DEPARTMENT OF MECHANICAL ENGINEERING

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis

Wireless Connectivity for Hilti Power Tools with Bluetooth

Author:	Mingyong Xu
Supervisor:	Prof. Julien Provost
Advisor:	Dr. Michael Fuchs
Submission Date:	6th october 2015

Acknowledgments

I would like to thank all my co-workers in their support with my work. Special thanks to Michael for all his advises for a better writing of this Thesis, to Greg for helping me with the STM module, to Roland for all his support and help for the tests, to Josip for all the works done as the intermediary with STMicroelectronics and to Francisca for letting me share my problems with her. And also thanks to my family and all my friends as they help me to overcome my difficulties.

Abstract

The increase use of ARM cores in industrial embedded systems make them more powerful. And as a method to implement a wireless communication we chose the relatively new Bluetooth Low Energy technology. In this Master Thesis we study the application and performance of Bluetooth technology on industrial power tools. We study and compare the performance of two Bluetooth modules for embedded solutions, one from STMicroelectronics and one from Microchip. And look if they are suitable as a wireless communication in industrial environment. We also study the energy performance and possibles ways to optimize it.

Contents

1. Introduction

Present industrial power tools and machines are no more stand-alone systems, they need to be connected to other electrical systems such as computers and smartphones. Those connections whether they are serial or parallel need a common ground signal reference. This is achieved through connecting the grounds of both systems. This kind of connections will not have any problem if both systems share the same power source or one of them is a peripheral that depends energetically from the other. But if each system is powered with different energy source then the connection can produce a ground loop fault because of possible difference in potential between the two grounds. The solution to avoid this fault is using the galvanic isolation.

One solution considered is the use of optoisolators or optocouplers, but this solution presents two constrains: it need a physical media and it have a relatively low baud rate, typically around 9600.

Then, the increase use of high capable and multifunctional ARM cores in embedded systems and the widespread of smartphones and Bluetooth capable devices, specially with the recent adoption of Bluetooth Low Energy, make us consider the use of Bluetooth as a communication interface. This would make possible the communication between embedded systems and smartphones, and also would make the embedded system more App friendly. But as a wireless solution, the security is a very important aspect to be considered. We will see if there are any security threads for Bluetooth technology.

In this thesis we will compare two Bluetooth modules designed for embedded solutions to see which one is more suited for industrial environment applications and working with power tools. The details and the criteria used will be explained in

Methodology. The objective of this thesis is to search a empirical method to select the more suitable Bluetooth module for possible future applications. And as for the approach, this thesis will be addressed to generic embedded developers, so we will not dive deep into the physical and link layer part of the Bluetooth specification.

With the increase of computing power and availability of smartphones it is worth considering not only the communication between industrial machines and computers or laptops but also the communication with smartphones. That is why in this research we will focus more on Low Energy because one of the modules can only operate in low energy mode and also to be energetically efficient for the smartphones. Other consideration is the strict regulation of Bluetooth Classic technology for iOS use.

2. Bluetooth overview

2.1. Introduction

Bluetooth technology was created by Ericsson in 1994 as a alternative to RS-232 serial communication interface. At present, Bluetooth Special Interest Group (SIG) owns the trademark and oversees the development of Bluetooth technologies. Since the first mobile phone implemented with Bluetooth in 2000, the Bluetooth technology spread across all kind of devices, specially with the adoption of the Enhanced Data Rate (EDR) in 2004 and High Speed (HS) in 2009. Later in 2010 Bluetooth SIG adopted the Core Specification v4.0. In this specification Bluetooth SIG incorporated two new features, the Attribute Protocol (ATT) and the new Bluetooth Low Energy implementation. The Low Energy (LE) technology is marketed as Bluetooth Smart [1].

Bluetooth technology operates in the unlicensed industrial, scientific and medical (ISM) radio band from 2.4 to 2.485 GHz, and uses Adaptive Frequency Hopping (AFH) between frequency channels of 1 MHz to reduce interference from other ISM band wireless technologies and constant electromagnetic noises. It creates a wireless user group network called *Piconet*. Figure 2.1 and Figure 2.2 are pictorial representations of the piconet topology in classic and low energy respectively.

This network is formed by connected devices within the same physical channel and resembles a star network topology where the central node acts as master and the peripherals act as slaves. For the control of the communications, there are low level protocols to handle all the configurations and parameters. These protocols divide into two groups, the low hardware level of physical link and the upper software level of logical link. Above those layers there are Bluetooth SIG defined *Profiles* (See Figure 2.3).

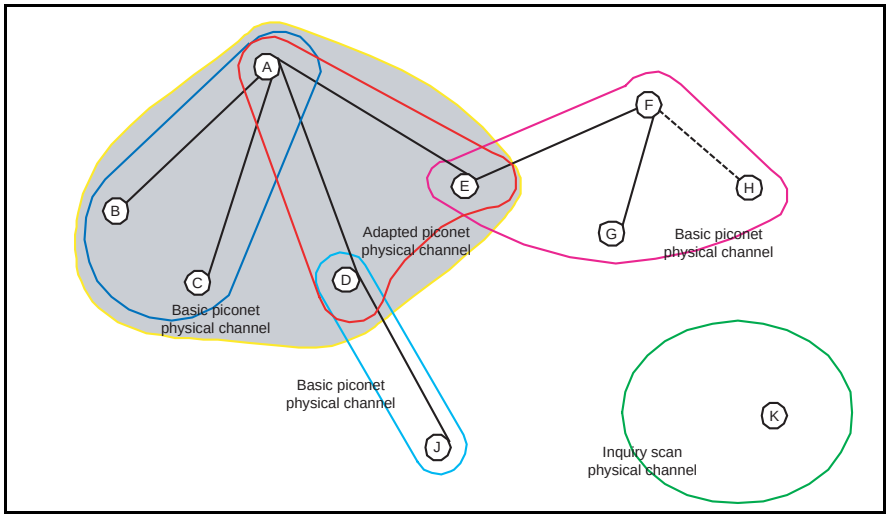

Figure 2.1.: Piconet topology for Bluetooth classic technology [2]

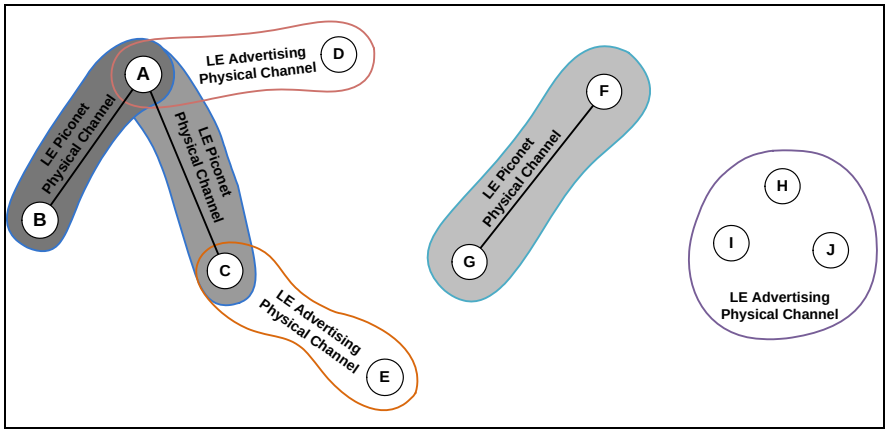

Figure 2.2.: Piconet topology for Bluetooth Low Energy [2]

The profiles are predefined configurations of the different layers and were thought for specific applications like hand-free, serial port, human interface, among others. At the end and above all there is the Generic Access Profile (GAP). The use of this profile is to describe the profile roles, connections and to control of the lower levels. It also configures the user faced aspects like Bluetooth device MAC address (BD_ADDR),

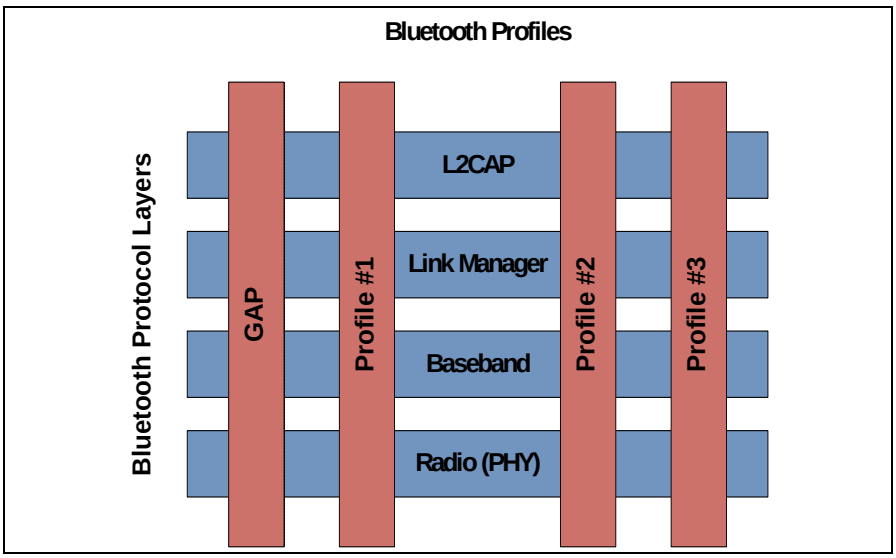

Figure 2.3.: Bluetooth Profiles and low level Protocol Layers [2]

Bluetooth device name, Bluetooth PIN, Class of Device and the pairing. All this protocol layers and profiles form the Bluetooth Stack [2] [3].

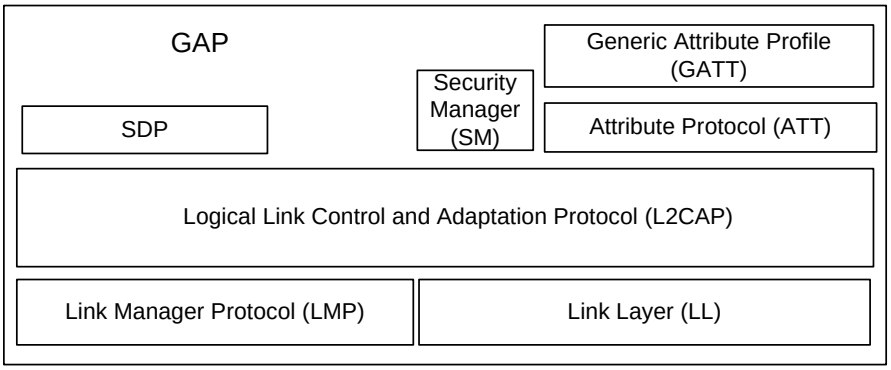

Figure 2.4.: GAP and lower layers defined in Bluetooth v4.0 [2]

The communication between the upper layers (Host) to the lower hardware layers (Controller) are handled by the Host Controller Interface (HCI). This interface defines a series of HCI Commands to access to the baseband commands, link manager commands, hardware status registers, control registers and event registers. Between Host and

Controller there are several intermediate layers. In order to transport the command through these layers the HCI uses the Host Controller Transport Layer, that carry the commands to the controller and notify the Host of HCI events. The transport layer provides transparency for the Host and intimacy of the data transferred. The whole architecture of the Host Controller Interface can be seen in the Figure 2.5.

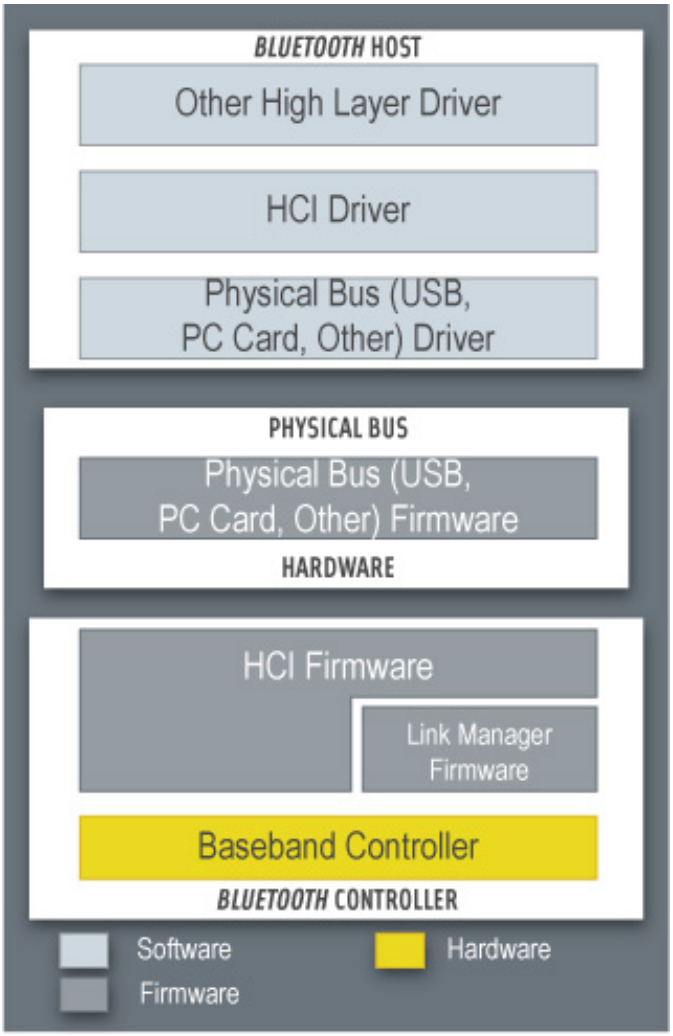

Figure 2.5.: Host Controller Interface (HCI) Architecture [4]

2.2. Connection procedure

The transmission of data occurs only in *Connection Events*, and in each connection event there is at least one packet sent from Master. Both Master and Slave send data alternatively, and even when the CRC is incorrect both must send back a packet. When the CRC is incorrect multiple times, then the connection event is closed. The connection event is defined by the connection event interval (*connInterval*) and slave latency (*connSlaveLatency*).

The connection event interval is the time between each connection event and it is started by the Master. The slave latency is the number of connection events that the slave can ignore. For situations like device moving out of range, encountering severe interference or a power failure condition there is *Supervision Timeout*. This timeout define the maximum time between two valid packets. Both Master and Slave must have its own supervision timeout, and when this timeout is reached the connection is closed.

2.3. Attribute Protocol (ATT)

The new Attribute Protocol was designed specifically for Bluetooth Low Energy, and will serve as base to the Generic Attribute Profile (GATT). In this protocol it is defined two roles, a server with a set of attributes and a client that will access and use this attribute. Each attribute have three unique values:

- an attribute type defined by an Universally Unique Identifier (UUID)
- an attribute handle, as a reference to identify the attribute in the server
- a set of permissions defined by higher layers

The UUID is a 128-bit value use in software standards to identify things, in this case an attribute. Because of its length it is hoped to be unique for all things, but there is still a probability of collision. For Bluetooth technology, there is a method to convert the 128-bit UUIDs to shorter 16-bit UUIDs and vice versa for frequently used UUIDs.

The attribute handle is a 16-bit values and is set internally by the server, this handle

must be unique and non zero. The handle is sent to the client so the client can reference to the attribute. The attribute values is a byte array that can be fixed or variable length. The length of this value is usually defined as less than the maximum length of the transfer packet. The default maximum transfer size for Bluetooth Low energy is 23 bytes. Although there is defined a Long Attribute Value much larger than the normal transmission size, in this cases the value must be divided into smaller packets in order to send it through Bluetooth. The maximum size of an attribute value is 512 bytes.

The attribute permissions are used to define the security level of the attribute. The attribute permissions are a combination of access permission, authentication permissions and authorization permissions.

The client sends attribute protocols requests to the server to obtain all the attributes and the server must respond to all of them. The attribute protocol has notification and indication capabilities to send attribute values to a client without the need for them to read it.

2.4. Bluetooth Low Energy (BLE)

Bluetooth LE was designed for devices that uses small or coin-cell batteries and be able to work for more than one year without changing it. It also operates in the ISM band and uses technologies like AFH, but there are many changes from the Bluetooth EDR and HS. Some of those differences are:

- a wider channel of 2 MHz
- an increase of modulation index
- use of a different network topology
- use of very short, up to 27 octets, data packages and a different packet format
- an increase of access address size to 32-bit, used to identify the device inside the piconet
- implementation of AES encryption

- use of the new Generic Attribute Profile (GATT)

This new Bluetooth Profile is designed specifically for LE. In the GATT Profile there are collections of data associated with particular functions or features. Each of this collections is called *Service* and there are two types of services, primary and secondary. The primary services are discoverable while the secondary only serve to be referenced or included by a primary. This service is an attribute type and has as value a handle to the *Characteristics*. A characteristic is also an attribute type, and its definition contains a declaration, properties like read/write access and the value that is the main part of a characteristic. Sometimes they may have some descriptors for configuration propose [2]. Figure 2.6 shows a graphical representation of the hierarchy of the GATT Profile.

Some of the GATT profiles and services adopted by Bluetooth SIG are:

- Blood Pressure Service
- Device Information Service
- Environmental Sensing Profile
- Health Thermometer Profile
- Heart Rate Profile
- Proximity Profile

The profiles define the service definitions and characteristics behaviours of the GATT Profile. Each of the services have its own UUID value defined by Bluetooth SIG in Assigned Numbers. Those UUIDs are the 16-bit short version of the full 128-bit UUID. When a company wants to use a service that are not adopted by Bluetooth SIG there is an alternative option of using private custom UUIDs. The Bluetooth SIG give to the companies who ask it a 16-bit UUID and it will be included in the 128-bit UUID used by the companies for their custom applications and services of Bluetooth Low Energy [5].

In classic technology the profile must be implemented in the Bluetooth Stack in both sides in order to be able to use it. Differently, in Low Energy technology all the profiles and services are only defines one-sided, generally the device that acts as a slave. This

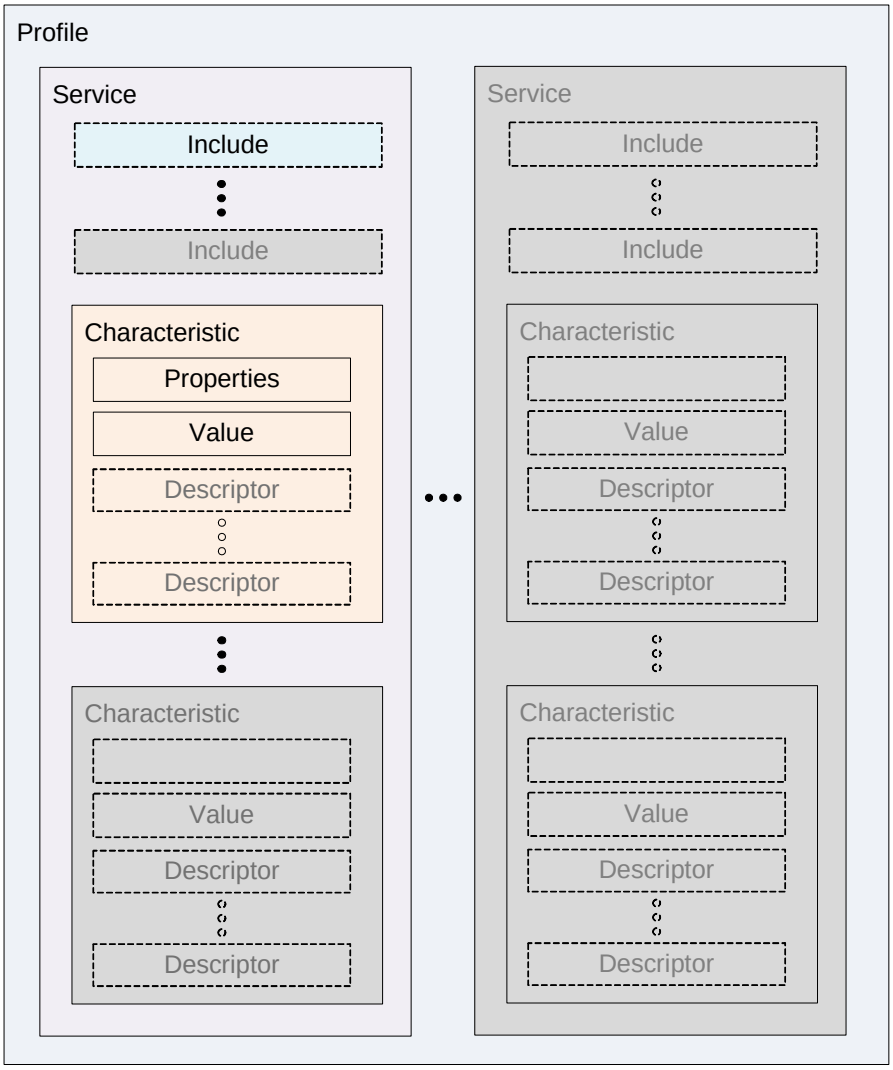

Figure 2.6.: GATT-Based Profile Hierarchy. [2]

device is called server, and the one that connects to it to use the services is called client. The client is the one who initiate the communication, so he acts as the master.

Because of those differences, LE is treated as a new Bluetooth type, and the technologies before LE are called by Bluetooth SIG as Classic. Although LE and Classic both use GAP, they are mutually incompatible as they cannot detect each other. This leads to two types of Bluetooth implementation, single-mode and dual-mode. In a

dual-mode implementation, Bluetooth Low Energy is combined with a existing Classic Bluetooth controller, so that it can use both LE and Classic technology [6].

All the services and characteristics can be obtained by the Service Discovery Protocol (SDP), in order that the client can use the services. So in default all the services and characteristics are open and can be accessed by any Bluetooth Low Energy device. If someone want to implement a secure and hidden service that are not available for general public and devices he can use the White List function in the Host Controller Interface (HCI) specification. This white list function permits to set discoverability and connectivity of Bluetooth devices depending if they are or not in the white list. This is useful specially when both Bluetooth devices are from the same company and are for the same application. Extra security features are allowed for vendor specific functions and Bluetooth Stacks.

The demand for Bluetooth LE accessories and applications have increased in the last years with the rise of the smartphones and small peripheral tools like wearable devices. Those devices that are able to use both the Bluetooth Low Energy and Classic technology are labelled as Smart Ready. Bluetooth SIG states that by 2018 more than 90% of Bluetooth enabled smartphones are expected to be Smart Ready [7].

3. Literature review

3.1. Performnce tests and applications

From the beginnings of Bluetooth technology there were already papers about Bluetooth application in industrial environment, like the one from Baños [8]. In his paper he designed a test system for different Bluetooth requirements in industrial environment. Those requirements are grouped in three levels, the basic radio frequency tester, the medium protocol tester and the upper lever of profile or application tester. The radio frequency and protocol tests are handled by Bluetooth SIG and the Governmental laws in regard of radiation of electromagnetic waves like the European ETS [9].

And with the rise of the smartphones, there is also a implementation for a real-time communication between an Android smartphone and a FPGA [10]. In his application he used a Xilinx Spartan 3E FPGA to connect to a Google Nexus using the Serial Port Profile of the classic technology. The Bluetooth module used is the PmodBT from Digilent and similarly to the RN4677 from Microchip it sets a transparent UART communication between those two devices. As the functionality of the Bluetooth module PmodBT is very similar to RN4677, the architecture defined in Figure 3.1 will be referenced to define the system architecture for RN4677.

In regard of the influence of the distance in Bluetooth performance, this paper [11] mentions the effect in a communication between a physical network servers and mobile devices using Bluetooth classic implementation. They used USB Dongle cards to implement Bluetooth in the server machines. In the results they observed that for distances below 10 m the throughput is quite steady, meaning that the influences of the distance in the performance are minimal for distances below 10 m.

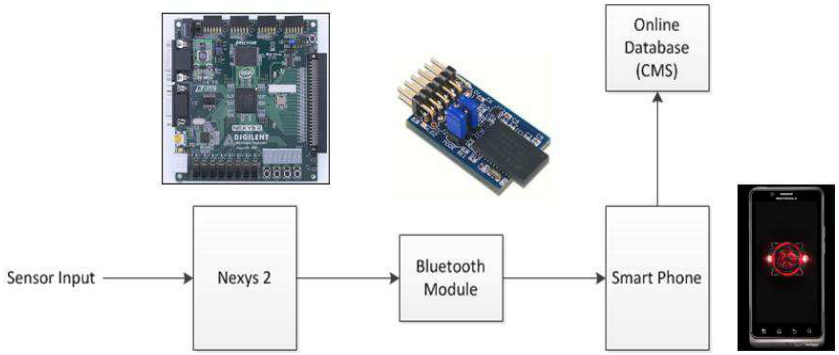

Figure 3.1.: System Architecture defined by [10]

3.2. Interference analysis

As the ISM Band is unlicensed there are many other wireless communications implemented in this band like IEEE 802.11/Wi-Fi, ZigBee and other wireless solutions for amateur embedded boards. This leads to an increase of interferences caused by collisions between this different wireless communications. In [12] a test is done to study the influences of WLAN and human body on the Bluetooth performance. The results are that the BLE frequency hopping is effectively avoiding the WLAN and classic Bluetooth, but the human body can absorb part of the emitted signal thus reducing the effective distance of the BLE signal. [13] also studies the interference between Bluetooth and IEEE 802.11b or WLAN. Unlike Bluetooth, which uses small packets and small frequency channels, the WLAN uses bigger packets in bigger channels in order to achieve a higher data bitrate. So the area of operation of Bluetooth is smaller than WLAN. At the end, this paper reached a similar conclusion as [12], it stated that "IEEE 802.11b transmissions are affected much more from Bluetooth, than Bluetooth suffers from the 802.11b".

Now we know that the influence of WLAN on the performance of Bluetooth is minimal, then we should study the other scenario, the impact of other piconets. As the number of Bluetooth devices increase year after year, specially now that more people

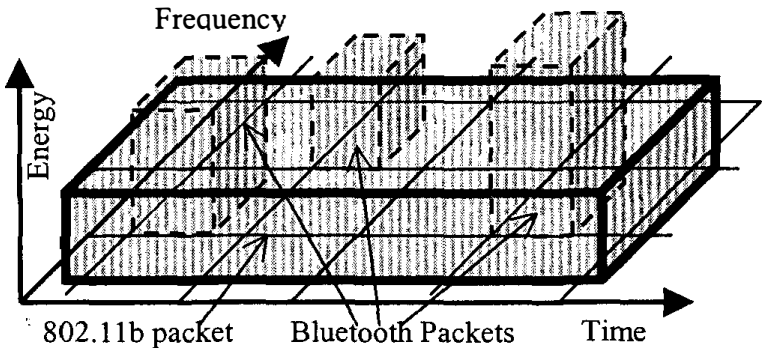

Figure 3.2.: Collision scenario between Bluetooth and WLAN [13]

have smartphones connected to Bluetooth devices or wearables, it is more probable an increase of interferences because of the coexistence of independent piconets. There are also numerous researches on the impact of co-channel interferences and possible mitigations. In [14] it is presented an improved prediction of the packet collision effect in a multipiconet environment, and some graphs showing analytical and simulated results of the impact of number of piconets on packet error probability. The Bluetooth technology studied here is the classic one.

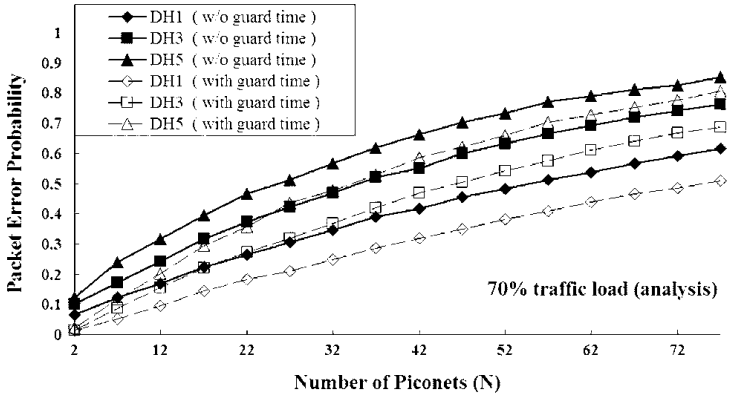

Figure 3.3.: Analycal results to evaluate the number of piconets effect on packet error probability [14]

In [15] it is also studied the impact of other Bluetooth networks on throughput performance, but this paper focus more on the effects of the packet length, defined

in the classic implementation, in the interference. He concluded that generally is preferable to use longer packets as the higher bandwidth efficiency compensates the collision problem, but at higher number of piconets it is better to use shorter packets as then could have more serious collisions. And that larger packets are more vulnerable to small packets for a similar reason as [13].

The authors in [16] study the effects of electromagnetic interferences from radio frequency emissions sources and from electrical and electronic equipments. The results can be seen in Figure 3.4, which shows the expected maximum distance between transmitter and receiver with a given signal to noise ratio and a given transmitter power.

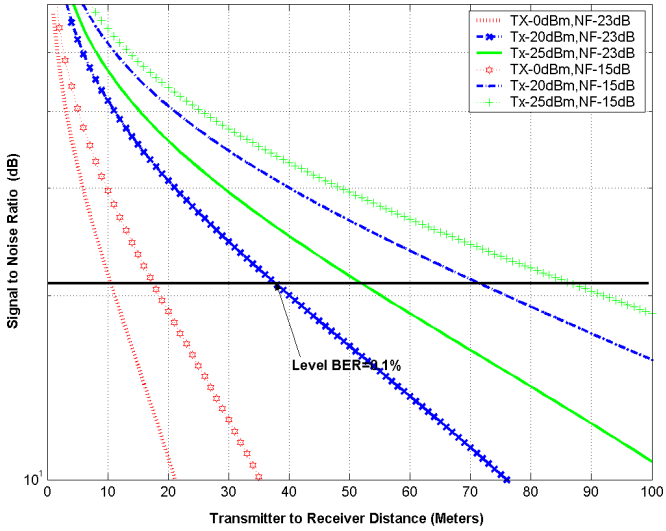

Figure 3.4.: Signal to noise ratio versus transmitter to receiver distance [16]

3.3. EMI from DC and AC brushless motors

Another source of interferences in an industrial environment is the electromagnetic interference (EMI) from electrical drives. The two electrical motors used to test the

performance of Bluetooth are a brushless DC motor and a brushless switched reluctance AC motor. Studies about the EMI from brushless DC drives are conducted in [17]. The results of this paper shows that the frequency of the EMI is very low compared with Bluetooth, it range between 300 Hz and 1000 Hz, but the power of such emissions are quite strong, it can rise up to 27 dB.

In both [18] and [19] they measured the EMI from different brushless switched reluctance AC motor connected to the power grid of 220 V and 50 Hz. The results differ from the case with DC motors, the noise spectrum have higher frequencies and also higher magnitudes. In this case, the EMI varies from 1 MHz to 30 MHz and the emission magnitude can rise up to 112 dBuV. Although the frequency is still far from the Bluetooth ISM Band, the intensity of the emissions is enough to be taken serious. The authors of [18] described in the conclusion that this EMI noise can be shielded with a metal shell.

3.4. Bluetooth Low Energy security

Because of the increase use of Bluetooth technology more and more people are looking ways to gain information in an unauthorized manner. As stated in [20], [21] and [22] there are many known security threads in Bluetooth technology. The author in [22] made a collection and classification of known Bluetooth attacks shown in Table 3.1.

Attack Classification	**Threat Level**	**Threats**
Surveillance	*Low*: Main purpose is to observe and gather information about the device and its location	Blueprinting, bt_audit, redfang, War-nibbling, Bluefish, sdptool, Bluescanner, BTScanner
Range Extension	*Low*: Main purpose is to extend the device range so that attacks could be conducted from far way distance	BlueSniping, bluetooone, Vera-NG

Continued on next page

Table 3.1 – *Continued from previous page*

Attack Classification	Threat Level	Threats
Obfuscation	*Low*: Main purpose is to hide the attacker's identity.	Bdaddr, hciconfig, Spooftooph
Fuzzer	*Medium*: Main purpose is to submit non standard input to get different results.	BluePass, Bluetooth Stack Smasher, BlueSmack, Tanya, BlueStab
Sniffing	*Medium*: Main purpose is to capture the Bluetooth traffic in transit.	FTS4BT, Merlin, BlueSniff, HCIDump, Wireshark, kismet
Denial Of Service	*Medium*: Main purpose is to deny resources to a target by saturating the communication channel.	Battery exhaustion, signal jamming, BlueSYN, Blueper, BlueJacking, vCardBlaster
Malware	*Medium*: Main purpose is to carry out attacks typically using self replicating form of software.	BlueBag, Caribe, CommWarrior
Unauthorized Direct Data Access	*High*: Main purpose is to gather private information in an unauthorized manner. This is very serious as very important information can be stolen.	Bloover, BlueBug, BlueSnarf, BlueSnarf++, BTCrack, Car Whisperer, HeloMoto, btpincrack
Man In The Middle	*High*: Main purpose is to place a device between two connected devices. All the information sent through the channel can be accessed.	BT-SSP-Printer-MITM, BlueSpooof, bthidproxy

Table 3.1.: Collection and classification of known Bluetooth attacks [22]

As it can be seen, there are lots of security threats in Bluetooth technology. But

as stated in [22] and [23], most of these threats are caused by the legacy 4-digit PIN pairing, that have been solved in the new Bluetooth v4.0.

The Secure Simple Pairing is a pairing method introduced in Bluetooth v2.1 for BR/EDR and use a 16 alphanumeric PIN and Elliptic Curve Diffie Hellman (ECDH) public key cryptography to prevent from threats up to unauthorized direct data access as passive eavesdropping. But it can still be vulnerable to man in the middle attacks. For those type of attacks secure simple pairing uses four association models designed for different input/output capabilities of the Bluetooth devices. Those models are Just Works, Out of Band and Passkey Entry; from which except Just Work model, all others provides protection against man in the middle attacks. [2, 22].

As for LE, it have some differences in security with BR/EDR. LE does not have Secure Simple Pairing but uses some of its association models, although the quality of the protection is not the same as in Secure Simple Pairing. The association models that LE uses are Just Works, Out of Band and Passkey Entry. Differently than Secure Simple Pairing, LE Just Works and Passkey Entry do not use Elliptic Curve Diffie Hellman, leaving them vulnerable to passive eavesdropping. In LE the keys is generated by the Host so it can be upgraded easily without changing the Controller, and each key have its own purpose:

- Confidentiality of data and device authentication
- Authentication of unencrypted data
- Device Identity

The encryption in LE adopts AES-CCM cryptography using a AES-128 bit block cypher and it is performed in the Controller. There are two additional security features for LE: send signed data from a trusted device over unencrypted ATT bearer and use of private Bluetooth address that changes frequently making it difficult to track.

All this security features are configured in the new Security Manager Protocol in Bluetooth LE GAP shown in Figure 2.4. In the new Security Manager Protocol the security requirements are configured in two LE security modes: *LE Security Mode 1* that provide security at the Link Layer and *LE Security Mode 2* that provide security at the

Attribute Protocol Layer. Each Security mode considers different security levels and configurations (shown in Table 3.2) [2].

	Mode 1			**Mode 2**	
	Level 1	**Level 2**	**Level 3**	**Level 1**	**Level 2**
Authenticated pairing	no	no	yes	no	yes
Encryption	no	yes		—	
Signed data	—			yes	

Table 3.2.: LE Security modes and levels [2]

Another aspect worth concerning is the fact that all the services and characteristics definitions does not require authentication or authorization to be read. This can cause some privacy problems as the developer can not hide the services and characteristics to his own. Fortunately there are options to set access permission to the characteristic values from higher levels, but it is not specified in the Bluetooth Core Specification.

3.5. Energy performance of Bluetooth LE

As explained in Chapter 2, the Bluetooth Low Energy technology was designed to work with a coin battery for years. But as explained in [24] this lifetime expectancy is dependent of the parameters *connIntercal* and *connSlaveLatency*. The effects are presented in Figure 3.5, from that we can extract that for longer lifetime or lower energy consumption we need a higher connection interval and higher slave connection latency.

In another paper is studied the energy consumption of device discovery in Bluetooth Low Energy [25]. They first studied the effects of different advertising intervals and scan intervals in the energy consumption of Bluetooth Low Energy. The results can be seen in Figure 3.6, where it is studied the different advertisement intervals from the server with the client set with a specific scan interval, and Figure 3.7, where the situation is inverted and the study is from the point of view of the client. From the graphics we can see that the server or advertiser needs a client with low scan interval, and the client or scanner needs a lower advertisement interval. So for an optimized

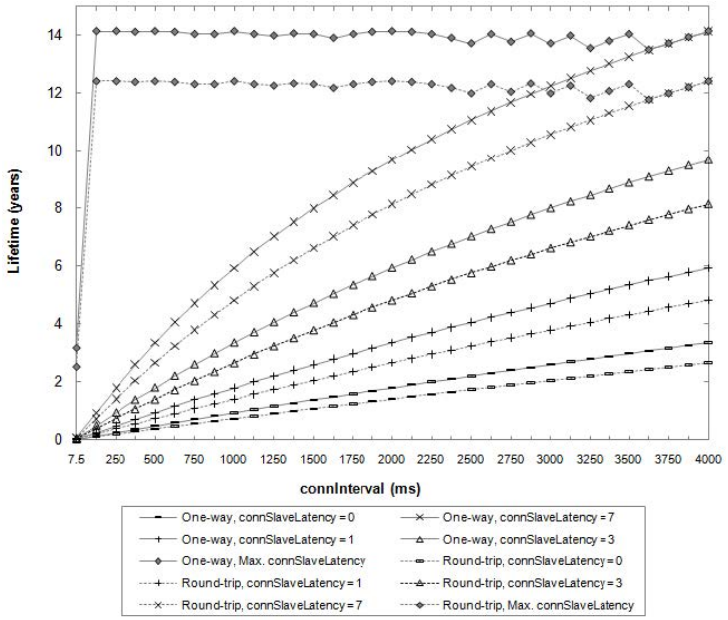

Figure 3.5.: Theoretical lifetime of a slave for one-way and round-trip ATT message exchanges, and for different parameter configurations [24]

energy solution we need to find a trade-off between the two devices.

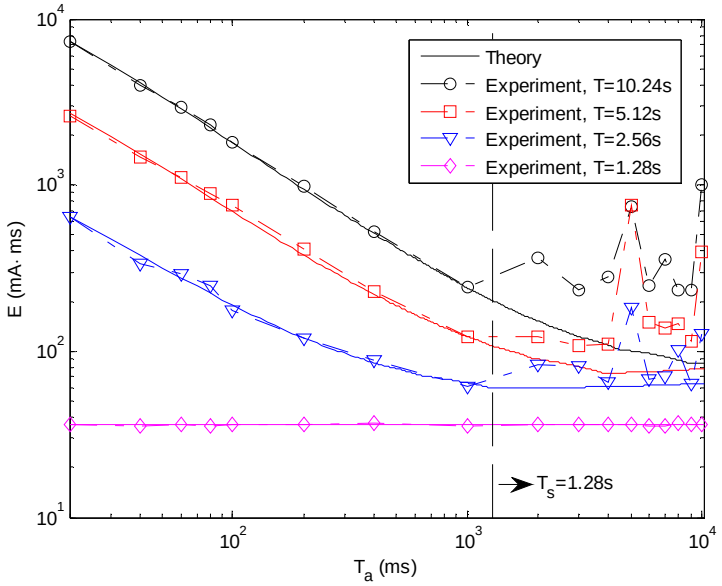

Figure 3.6.: Average energy comparison with varied T_a (advertisment interval) in different T (scan interval) situations [25]

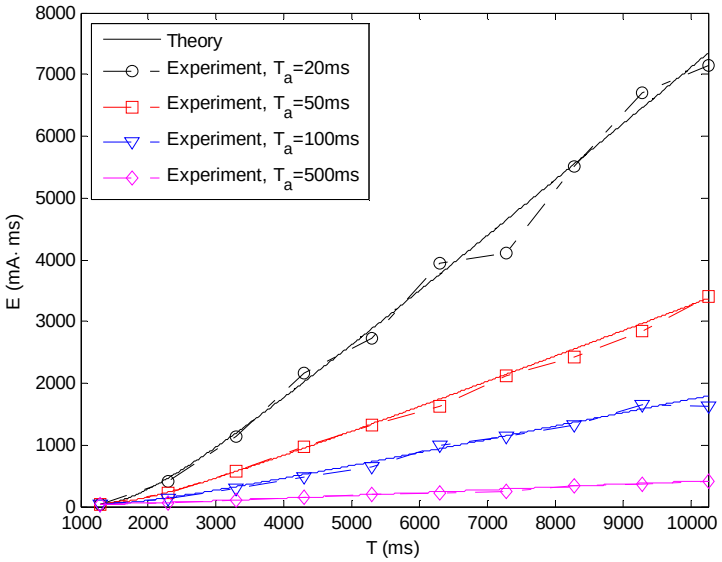

Figure 3.7.: Average energy comparison with varied T (scan interval) in different T_a (advertisment interval) situations [25]

4. Methodology

4.1. Architecture and design

As said in the Introduction this thesis is mainly focused in the application of Bluetooth technology in embedded systems, so the main point of study will be the actual application performance. In that way we will not study the Bluetooth signal performance and logical algorithms optimization. Also because the Bluetooth modules tend to be black boxes so the internal firmware of the module can not be changed or optimized.

The Bluetooth modules that will be studied and compared are the RN4677 from Microchip Technology Inc. [26] and the BlueNRG from STMicroelectronics [27]. The main reason for choosing this two companies is because the target embedded system uses a ARM Cortex-M made by STM and the Microchip module is relatively easy to implement.

The RN4677 is a dual-mode Bluetooth module with a UART interface and have a internal high abstraction level firmware. This firmware controls all the Bluetooth communication protocols and configurations, so that from the UART interface perspective the module is transparent. In Classic mode it is configured automatically for using the Serial Port Profile (SPP).

BlueNRG is a single-mode Bluetooth Low Energy network processor and communicates with the embedded system by a Serial Peripheral Interface (SPI) connection. But unlike the RN4677, BlueNRG have a lower level Bluetooth API, so all the communication and settings must be done manually through an application controller interface (ACI) based on SPI. This is done through a list of commands for both HCI and GAP/GATT.

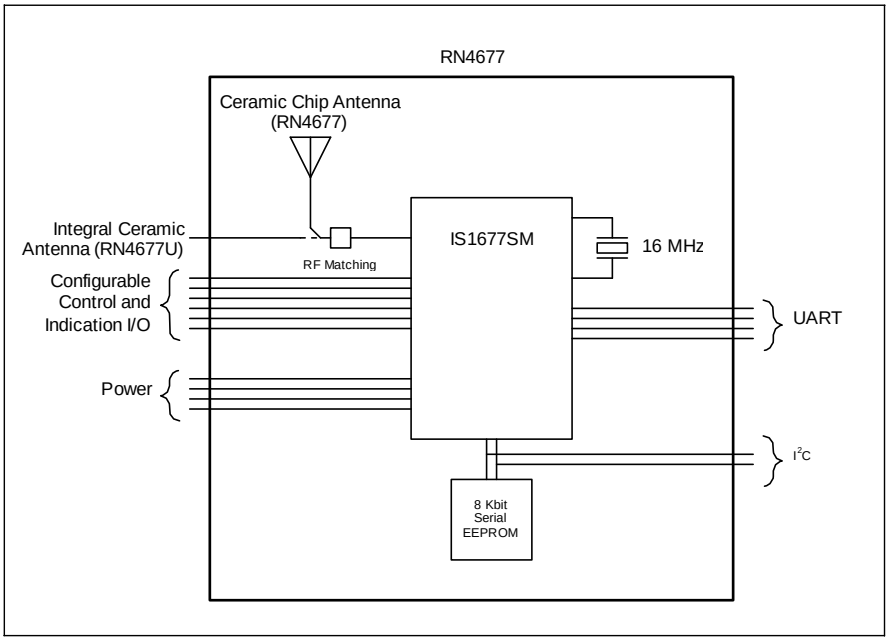

Figure 4.1.: RN4677 electronic block diagram [26]

Because of the physical and software limitations the tests will be performed on different chipsets. In the case of BlueNRG, because of the difficult portability of the libraries, we will study its performance in the expansion board X-NUCLEO-IDB04A1 [28] from STM attached to the development board NUCLEO-L053R8 [29]. The final module can be seen in Figure 4.3. And as for the RN4677, it will be connected to the test board MCBSTM32 with the microprocessor STM32F103. The difference in chipset will be considered in the analysis of the results.

For the serial communication we defined in the STM module a *Chat* service with the UUID given by STM *D973F2E0-B19E-11E2-9E96-0800200C9A66*. This service include two characteristics, these two characteristics simulates the TX and RX properties of a serial interface. Both characteristics is defined with a maximum length of 20 bytes and none access permission restrictions, in order to study only the communication performance and not the influences of the security manager.

The TX characteristic, with the UUID *D973F2E1-B19E-11E2-9E96-0800200C9A66*, is

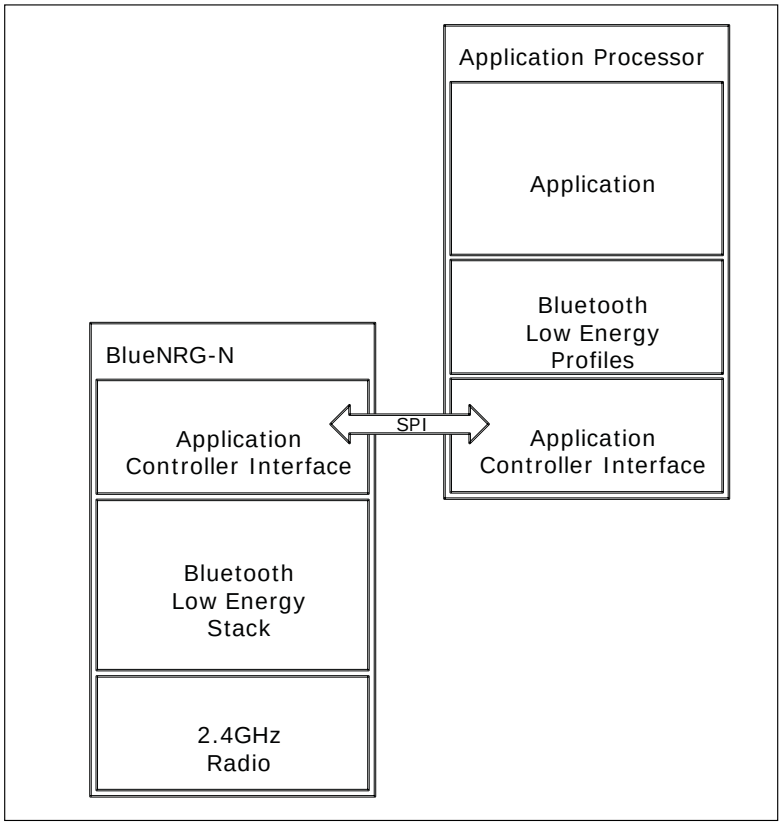

Figure 4.2.: BlueNRG application block diagram [27]

the transmitter of the server with notification enabled. That means the server will write data in TX value and the characteristic will notify the connected client a modification of this value.

The RX characteristic with UUID *D973F2E2-B19E-11E2-9E96-0800200C9A66* acts as the receiver of the server. The value is set to be writeable and when the connected client writes or modifies the value of the characteristic, the ACI will notify the server of a modification in the value and then call the interrupt subroutine defined by the user.

In the Microchip's RN4677 module there are already defined a similar service with the same TX and RX characteristics. We only needed to change the predefined UUID to the three we mentioned above.

Figure 4.3.: BlueNRG expansion board connected to STM32 Nucleo board [30]

The main test environment will be a closed industrial product development and testing laboratory with big machinery, power stations, transformers and wireless communications like WLAN. This setting can be ported to other industrial situations like a production chain, as the equipment and electromagnetic interference are similar. The objective of this approach is to study the effects of a real industrial situation in the Bluetooth communication and the capability of the modules to face against common industrial interferences. This noises will be treated as background constant noise to contrast with the dynamic punctual noise from electric motors, and will be present during all the tests.

4.2. Implementation

As a substitute of serial communication, our main variable of study is the speed of data transfer. The aim is to implement a wireless communication interface capable of high speed data transfers, because one application may be programming or updating the firmware of the embedded system, so from a service part it is important that it do not take long time to upload. As one of the Bluetooth modules is not compatible with classic technology the comparative of the modules will be mainly focused on the

performance of both LE applications. Additionally will be discussed the difference between the classical and the low energy.

Other variables considered that can influence the system will be the distance between the two devices, serial COM baudrate, the existence of other wireless communications in the same ISM band and the existence of other electromagnetic noises like the ones produced by a induction motor. If we study all the combinations possible for all the variables, then the test model can be relatively big to study. That is why the influences of each variable will be studied separately, using different scenarios to isolate the effects of the variables. For all the tests, will be considered the case of maximum payload, that is the case of packets of 20 bytes. This is to maximize the ratio between actual data and headers for the protocol.

The method used to study the performance of the communication is a round-trip chat between the client or master (in our case is the notebook PC) and the server or slave (in this case is the Bluetooth module connected to a embedded system). The computer will initiate the conversation and will measure the time needed to send a random generated data and to read the response from the server and this measured time will be used to study the performance of the communication. In the side of the server, in order to minimize the response time, the program implemented is only a mirror function that returns the data received from the computer as soon as it arrives. To make it as fast as possible, the mirror function will be called in the subprocess of the data income interruption. With the STM module the interrupt is called when the characteristic value is changed. In the case of Microchip module it uses the interrupt of incoming byte from the UART interface.

The C code for the mirror function and for the time stamps are attached in the Appendix A.

The Figure A.1 shows the subprogram used in Windows to get the time stamps in milliseconds in order to calculate the time between just before sending the data and just after receiving the response from the server.

The Figure A.2 is the implementation of the mirror function in the STM server. This function is called by the *EVT_BLUE_GATT_ATTRIBUTE_MODIFIED* event, which will

pass the data buffer containing the new attribute value and the length of it. Then this function will immediately write the changed RX value in the TX value.

The Figure A.3 and Figure A.4 are the two parts of the mirror function implemented for the MCBSTM32 Board. And will be called for each byte by the UART interface interrupt.

Because the computer we use have Windows 7 OS, it have a Bluetooth stack with only v2.1, so we are unable to program a application which can use Bluetooth low energy directly. The solution arrived is to use a USB dongle capable of Bluetooth low energy, STM offers two different boards that can act as a dongle, the STEVAL-IDB002V1 development board [31] and the STEVAL-IDB003V1 [32]. Both USB dongles have a BlueNRG module and an external STM32L series low energy microcontroller. The program compiled is a little variation of the default chat application, it uses a virtual serial COM interface on the USB side and send the data received from the virtual COM port to the predefined Bluetooth low energy service and characteristic. So it works as a transparent Bluetooth serial communication for the computer, the computer only need to send the data through the virtual COM port created.

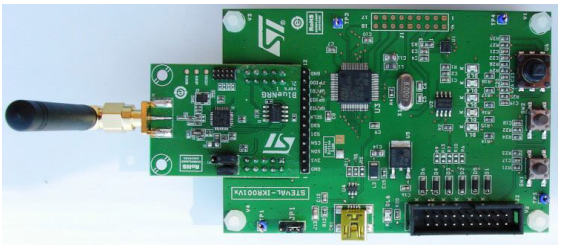

Figure 4.4.: STEVAL-IDB002V1 Bluetooth low energy board [31]

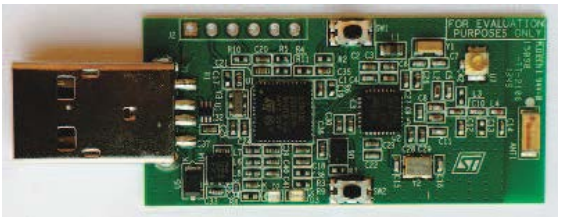

Figure 4.5.: STEVAL-IDB003V1 USB Bluetooth low energy dongle [32]

The two industrial tools that will be studied its electromagnetic interferences are from a power tools manufacturing company. One will be a DC motor tool and the other will be an AC motor power tool. The Bluetooth modules are positioned near the electrical motor in order to maximize its effects.

The DC motor tool uses a brushless high power DC motor that operates with a Li-ion battery of 36 V. As any brushless DC motor it uses a permanent magnet and inverters, generating two different back electromagnetic field, one from the permanent magnet and one from the poles or phases of the stator. And the AC motor tool uses a brushless switched reluctance (SR) AC motor connected to the grid power of 230 V and 50 Hz. And like any switched reluctance motor it does not use any permanent magnets rather a soft magnetic material making them having only one back electromagnetic field. From the researches shown in chapter 3, the probability of collision is practically none, but the strong electromagnetic emissions can influence the performance of Bluetooth devices as they tend to send relatively weak signal, around 4 dBm.

In the core specification is also defined a long attribute value longer than 20 bytes and a maximum length of 512 octets. In Bluetooth Low Energy there is a prepare write request and execute write request to write attribute values longer than 20 bytes. But there is no option in this protocol to know if an attribute value is longer than 20 bytes, it depends on the vendor specific implementation [2].

Also will be studied the RFID technology as a source of interference. The frequency band of the RFID system used is 860 MHz to 960 MHz, defined in the Part 6 of ISO/IEC 18000, and uses passive RFID tags. As the RFID system works out of the Bluetooth band, the interference caused by same frequency band will be minimum. So the it will be more like the study of interference from electrical motor, as a source of strong electromagnetic signal. The Bluetooth server module will be placed just above the RFID tag reader, while this later is constantly reading data from different passive tags.

In regards to energy optimization, as we prioritize the performance of data transfer and bit rate the connection interval will be set to minimum possible for both modules. The minimal connection interval for Microchip is 10 ms, and for STM is 7,5 ms. So both modules will be set to a minimal connexion interval of 10 ms.

5. Findings and analysis

5.1. Difficulties before testing

First of all, we wanted to see with which COM port baudrates we are able to work. But the computer creates a virtual COM port with the STM USB dongle to communicate with the other Bluetooth low energy devices, that means the baudrate or transfer speed depends on the USB driver on the dongle and the speed of the computer. The default baudrate of 115200 is coded inside the USB libraries and there is no option to be changed from outside the driver.

From the server side, the STM module uses the SPI and it let the developer to chose a frequency prescaler to set the baudrate. The Microchip module have also a parameter in the EEPROM where user can define the baudrate. But for baudarate above 115200 it start to have communication faults with the microprocessor. Because it have only one data flow control pin, the CLS, and with some tests on the CTS pin, we observe that the Microchip module does not answer to the RTS requests. So for all the tests the baudrate is set to 115200.

One of the handicaps found is the mentioned difficult portability of the BlueNRG library, the Profile Command Interface (PCI) and the Host Controller Interface (HCI). These main API are used to send GATT-based commands to the BlueNRG (Figure 5.1 shows a graphical representation of the software architecture). In the theory with the PCI and the hardware related libraries one should be able to port a Bluetooth application across all STM microprocessors, but in the reality there are still lots of compatibility problems between the PCI found in the Host Controller Interface (HCI) and the hardware libraries like Hardware Abstraction Layer (HAL). As it turns out

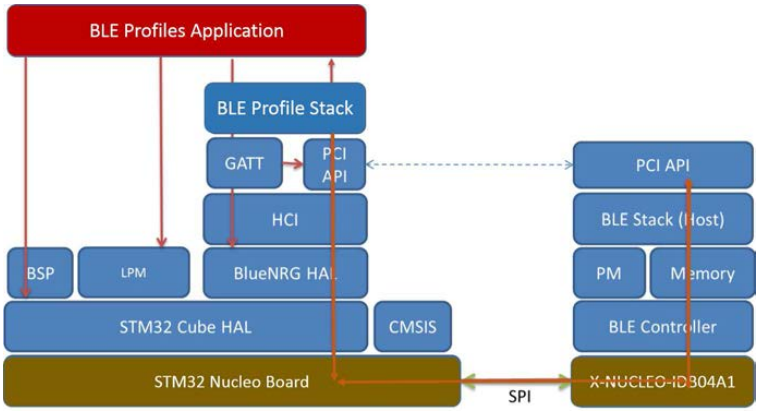

Figure 5.1.: Software architecture of BLE applications on STM32 Nucleo Board [33]

that the HAL has not the same abstraction level for all microprocessors from STM. That makes the HCI can only work with some specific microprocessors. That is why in order to compile a working application with Bluetooth we must use the examples applications for a specific development board.

Then, there is the compatibility of the HCI commands with the Firmware of the BlueNRG. All the new Development Kits comes only with the newest 6.4 Firmware for the BlueNRG, but the test projects were using the libraries designed for the older 6.3 Firmware. After reading through the libraries and testing, we concluded that the 6.4 Firmware is not backward compatible with 6.3.

This Bluetooth module is relatively new so we hope in the future they will solve this problem so we will be able to port or use this module with other microprocessors.

Some difficulties was also found in the Microchip module. Although we did not have library problems because of the high abstraction level of the module, we still had some difficulties. One of them is that we cannot change the configuration written in the EEPROM once the module is welded in the embedded system. There are some libraries for changing the configuration EEPROM during runtime, but it did not work as expected.

The other problem was that in order to write the configuration table to the EEPROM

we need a proprietary software and flashing tool. This dependency of third party tools can be annoying, specially for big companies.

The STM BlueNRG Bluetooth application command interface (ACI) [34] included in the PCI have some useful vendor specific commands. One of the vendor defined commands is the characteristic definition with security permissions. Unlike the whitelist option in the HCI that can only store a limited number of Bluetooth addresses, this security permissions allows write and/or read permission to any Bluetooth device that have been authenticated. This option have been test to work with smartphones but the client device have got problems trying to bond and authenticate with the server. If it works with the client device, it will add a additional layer of security besides the company internal command protocol.

Another of these commands is the *Aci_Gatt_Write_Long_Charac_Val* command that is used to write characteristic values longer than 20 bytes. Like defined in the core specification [2] this command divide a long data stream of maximum 255 bytes (value length of 1 byte) to smaller data blocks of 18 bytes and send it successively through Bluetooth. The STM module is able to acknowledge these data packets that come from this command and regroup them into a single attribute value. But the implementation does not work properly because it write down to the attribute value each time a data packet comes, unlike suggested in the core specification. So the application does not know which is the last packet or when does the process finish, when the incoming long data length is unknown. Also for unknown reasons, the server only receive a maximum of 64 bytes for whatever data length bigger than that. The bit rate for these 64 bytes has been tested to be around 4400 bits/s, far less than the normal 20 bytes transfer mode. Because of the results mentioned above, we decided that we will not use this command.

If in the future there is a need to write or read data blocks bigger than 20 bytes, it should be handled by the application layer with user defined protocols. As seen above, the split of the big data block into smaller packets is a viable solution.

5.2. Test results

5.2.1. Tests for the effects of the distance and electrical power tools

The first tests was performed with the STEVAL-IDB002V1 board, because the STEVAL-IDB003V1 USB dongle could not work because some compatibility problem with the firmware of the BlueNRG mentioned in section 5.1.

Figure 5.2 shows the first 9 scenarios tested. These tests were to see the influence of the distance and electrical motor on the performance of the both modules and in classic mode. We can appreciate these influences in how long last one round-trip chat, used as a response of the system. The Microchip module is set to transmit with a power of 2 dB and the STM module is set to -2 dB. The power set for Microchip is the maximum power available, while the power for STM is the default recommended by STM. Later the STM module will be tested with the maximum 8 dB to see if there is any significant difference. Each of these graphs can be seen in detail in the Appendix A.

In the top row we have the performance of the Microchip RN4677 module working in Classic Mode. First without any dynamic interference from electrical motors, and then with interference from the DC power tool and the AC tool. We can see that the three graphs are practically the same and with very little variance, meaning that there is minimum to none effect from both the distance and the electromagnetic noise. Also the mean line shows a steady and invariant performance over all.

The second row shows the performance of the same module in Low Energy mode. First of all we can see a higher mean value than in the Classic Mode. This means that overall the RN4677 is transferring data more slowly in low energy mode than in classic. We can also see a greater global variance of the response and an increase of packets with longer times in 15 m. Although, this later does not affect much in the mean value as it quite constant for all distances. There is a slightly difference between the situation without any tools and the one with the motors running, but this difference is hardly appreciable. So the electromagnetic effects from the motors have not a big impact on the transmission with Microchip module.

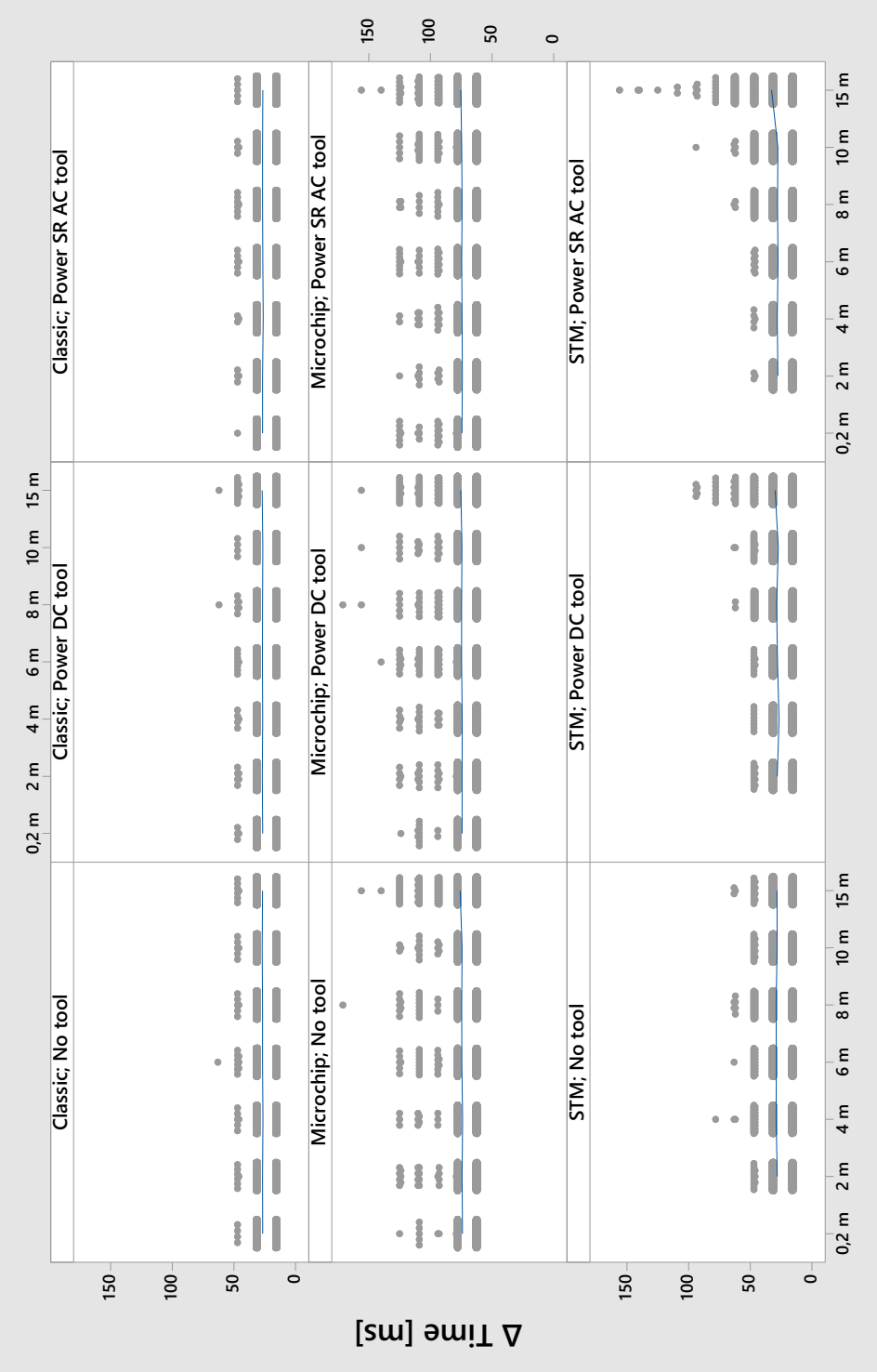

Figure 5.2.: Graph matrix of performance test

The last row descries the performance of the STM BueNRG module working in Low Energy mode. First it is clear that the mean values are lower that the Microchip counterpart, as also it is the variance. Up to 10 m there is no significant differences between the three cases, the main discrepancies is shown in the 15 m mark. We can see that because of the electromagnetic noise the number of round-trip chats with higher time increases hugely at this distance. As studied in section 3.3, despite the electromagnetic noise generated is far from the ISM band frequency, the power of the emissions is much stronger than the one we are transmitting. Specially in this case where we are sending packets with -2 dB. As the noise generated from an AC motor is more powerful than the DC motor. It is why the situation with the AC power tool the influence is much higher than the DC power tool. But the mean value does not change much from below 10 m to 15 m.

In all cases can be noted the stratification of the data, this is caused by the resolution of the Windows timers and connection interval for Bluetooth low energy. And the increase in time we mentioned before is a manifestation of the packets lost or corrupted by the interferences.

A collection of the mean bit rates for the different scenarios is presented in Table 5.1. These bit rates are calculated from the mean time values from Figure 5.2.

	Microchip RN4677			**STM BlueNRG**			**Classic Mode SPP**		
Tool	**No**	**DC**	**AC**	**No**	**DC**	**AC**	**No**	**DC**	**AC**
0,2 m	4323	4309	4300	—	—	—	12044	12038	11954
2 m	4304	4306	4320	11078	11433	11628	11996	11862	11967
4 m	4328	4312	4309	11064	12044	11418	12109	12002	12198
6 m	4296	4284	4290	11153	11511	11689	11979	11921	12051
8 m	4289	4282	4304	11068	11215	11508	11975	11977	11954
10 m	4310	4303	4289	11394	11797	11646	11837	12090	11947
15 m	4217	4245	4249	11364	10805	9809	11954	11856	12074

Table 5.1.: Mean bit rates of the different configurations in a closed environment

The Table 5.1 shows there are no direct dependence or influence of the distance and electromagnetic noise on the bit rate of the Bluetooth LE transmission. Although the effect is not totally null as could be seen in the graphs above, the impact on the mean bit rate is inappreciable. But for STM BlueNRG module operating with -2 dB it have still a slight drop on the bit rate for distances bigger than 10 m, specially when there are other external interferences. About the use of different microprocessor, we can see that the STM32F103 can achieve bit rates similar to STM32L151 in Classic Mode. This means that the bottleneck of the low bit rate of RN4677 in Low Energy mode is not caused by the microprocessor, but by the module itself.

5.2.2. Comparison between the two dongles

After we solved the problem with STEVAL-IDB003V1 flashing the right Firmware, we wanted to see if the two Bluetooth dongle are interchangeable. So we tested both devices in the same conditions and with the maximum transmission power, 8 dB. The results are shown in Figure 5.3 and Figure 5.4 respectively.

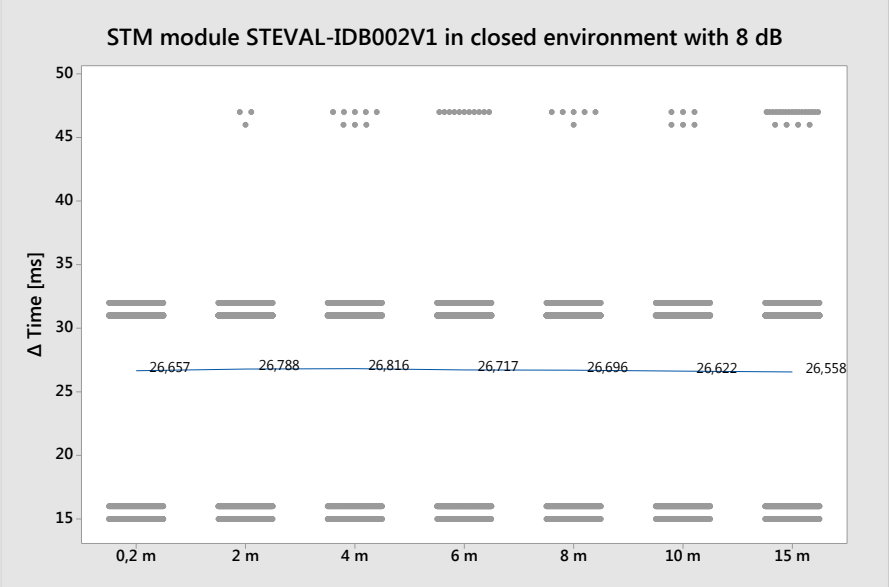

Figure 5.3.: Test with STEVAL-IDB002V1 development board with 8 dB

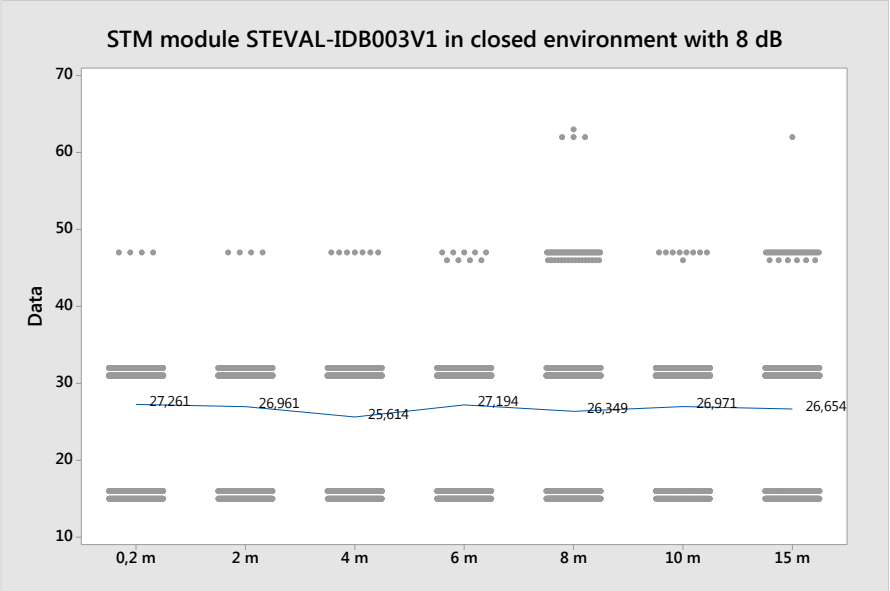

Figure 5.4.: Test with STEVAL-IDB003V1 USB dongle with 8 dB

As we can see, there is no significant difference in performance between this two devices. So they are completely interchangeable.

5.2.3. STM BlueNRG 8 dB testing

After we saw that the two dongles are basically the same we wanted to continue testing with 8 dB, to see if this high output power setting have any significant effects in the performance.

The outcome using 8 dB in the closed environment can be seen in Figure 5.6. It have clearly a better performance than when it was working with -2 dB.

About how it improve the performance facing the interference from a motor, we test it with the AC power tool which is the most influential of the two drives. Figure 5.8 shows the performance of the module in this setting. Although it truly improved the results, there is still a influence from the motor in 15 m. The 8 dB output cannot overcome the distance of 15 m while the switched reluctance AC motor is interfering.

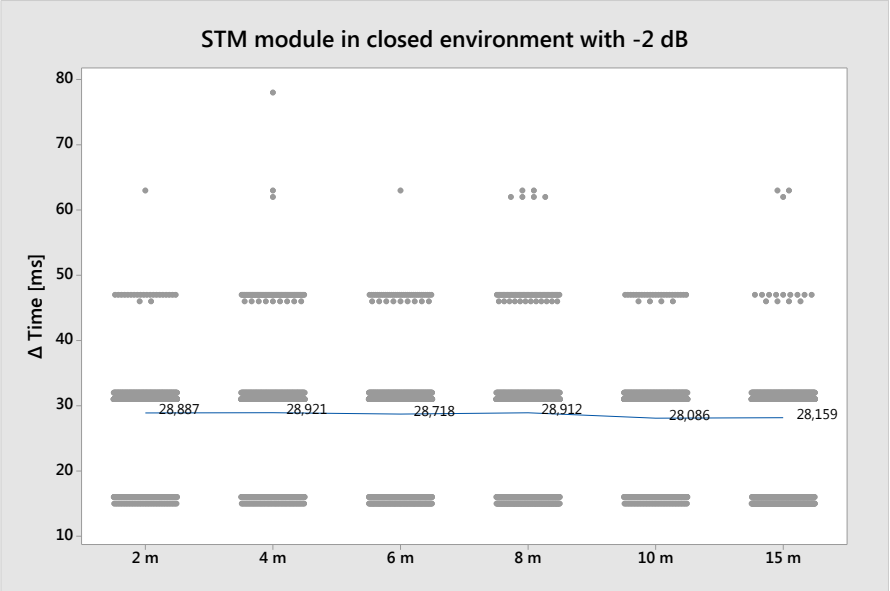

Figure 5.5.: STM module in closed environment with -2 dB

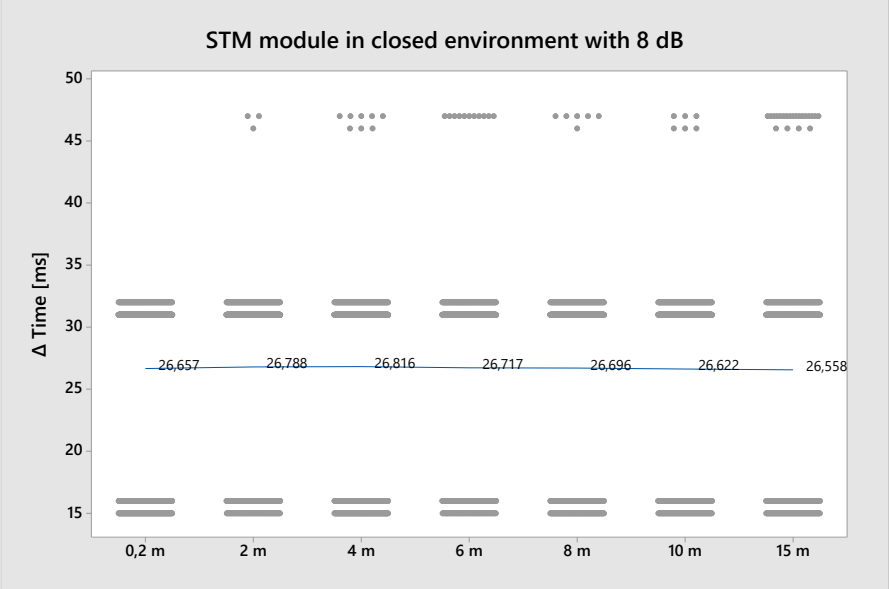

Figure 5.6.: STM module in closed environment with 8 dB

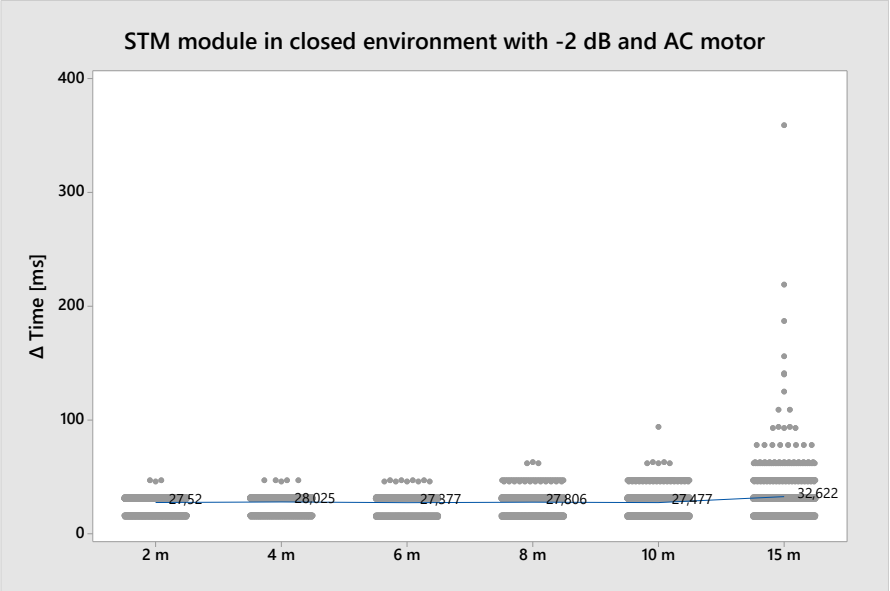

Figure 5.7.: STM module in closed environment with -2 dB and AC motor

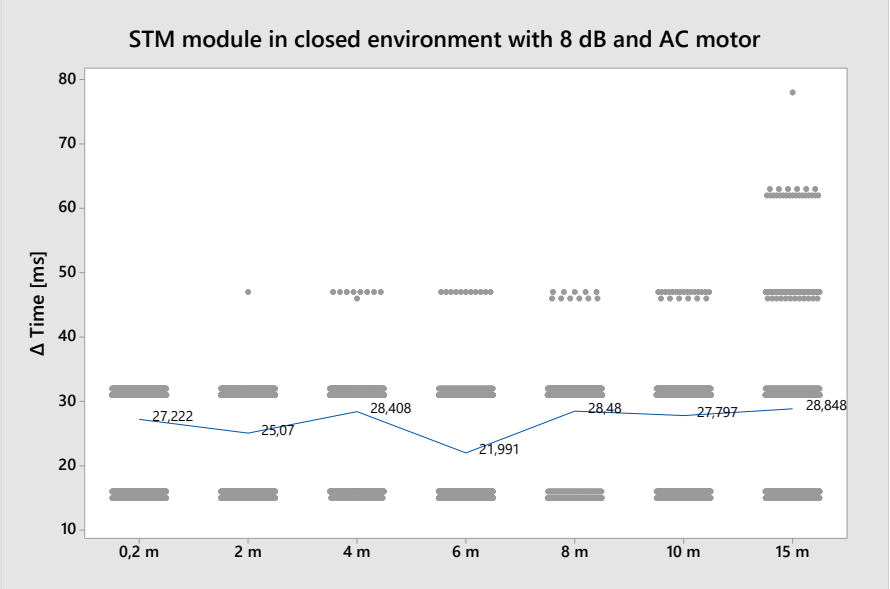

Figure 5.8.: STM module in closed environment with 8 dB and AC motor

The Table 5.2 shows the bit rate in the four situations. It is noted a slightly increase of the bit rate with the 8 dB transmission power, that is because it fights against the interferences better so it take less time to deliver the packets.

	No tools		**AC power tool**	
	-2 dB	**8 dB**	**-2 dB**	**8 dB**
2 m	11078	11946	11628	12764
4 m	11065	11922	11418	11264
6 m	11143	11977	11689	14551
8 m	11068	11987	11508	11236
10 m	11394	12020	11646	11512
15 m	11364	12049	9809	11093

Table 5.2.: Mean bit rates to compare the two levels of output power

5.2.4. Sending large data

The maximum size of a Bluetooth low energy packet is 23 bytes, if we want to send a data block bigger than that we need to split the block into smaller packets and send them consecutively. The performance can be improved because we will able to send more packets in the same connection interval or it can be worse as the errors are accumulated.

We tested first with the *Aci_Gatt_Write_long_Charac_Val*, but as mentioned before, it did not work as expected. As a alternative to the write long values, we tried to send a long data packet divided into short packets of 20 bytes and send it consecutively using the normal write command.

For this test we send consecutively 100 packets of 20 bytes random data and waited for the 20 bytes acknowledgement from the server. The Figure 5.9 shows the time needed for sending this 100 packets from different distances. It can be seen a increase of transmission time bigger than when we send packet per packet. This may be caused by the accumulation of the errors or longer send times we saw before.

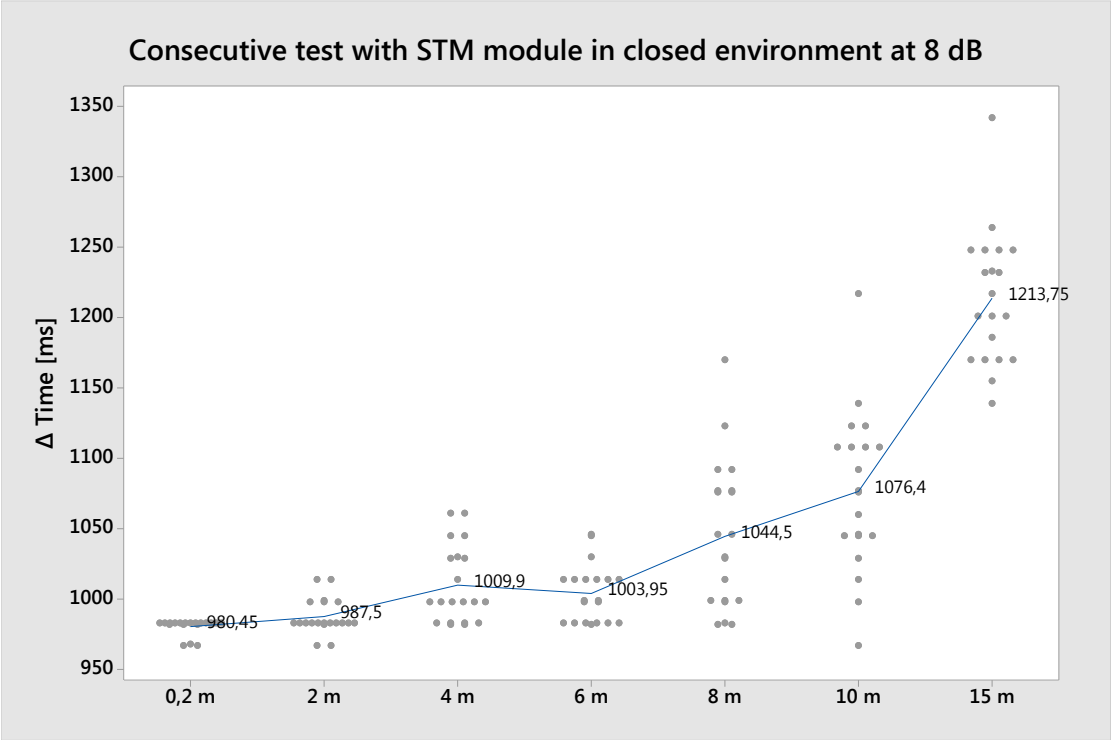

Figure 5.9.: Consecutive test with STM module in closed environment at 8 dB

	Simple packet test	Consecutive 100 packets test
0,2 m	12004	16482
2 m	11946	16365
4 m	11922	16002
6 m	11977	16096
8 m	11987	15472
10 m	12020	15013
15 m	12049	13314

Table 5.3.: Mean bit rates to compare the two levels of output power

To put it into perspective, we must compare the bit rates of the two transfer modes. This bit rates are calculated and showed in Table 5.3. It can be seen that in average the

bit rate of the consecutive mode is nearly 30% higher than the simple mode, despite the increase of the transmission time saw in the graphic above. So for futures uses it is recommended to send the data in this streamed consecutive mode.

5.2.5. RFID influences

The Figure 5.10 shows the response of the STM module facing the interference of a RFID reader described in chapter 4. As it can be seen, the difference can be considered none existent as predicted.

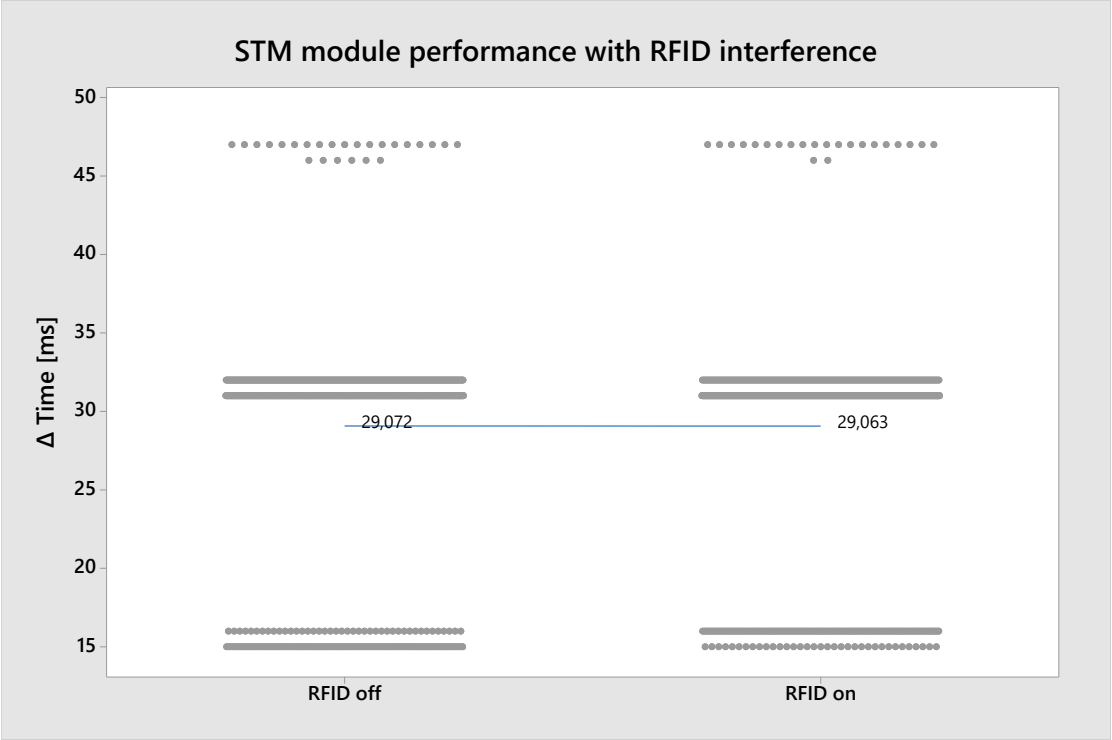

Figure 5.10.: STM module performance with RFID interference

5.2.6. Energy optimization

We saw in chapter 3 that for a longer lifetime we need to configure the connection interval parameter bigger. That is because the frequency of sending packets will be lower making it consume less energy. But that leads to a slower bit rate as there will be empty gaps between each packet.

In this section we will discuss which is the best trade-off between energy consumption and data transfer performance. The data collected are from the software *BlueNRG Current Consumption Estimation Tool* from STM. So all the data is theoretical and estimations.

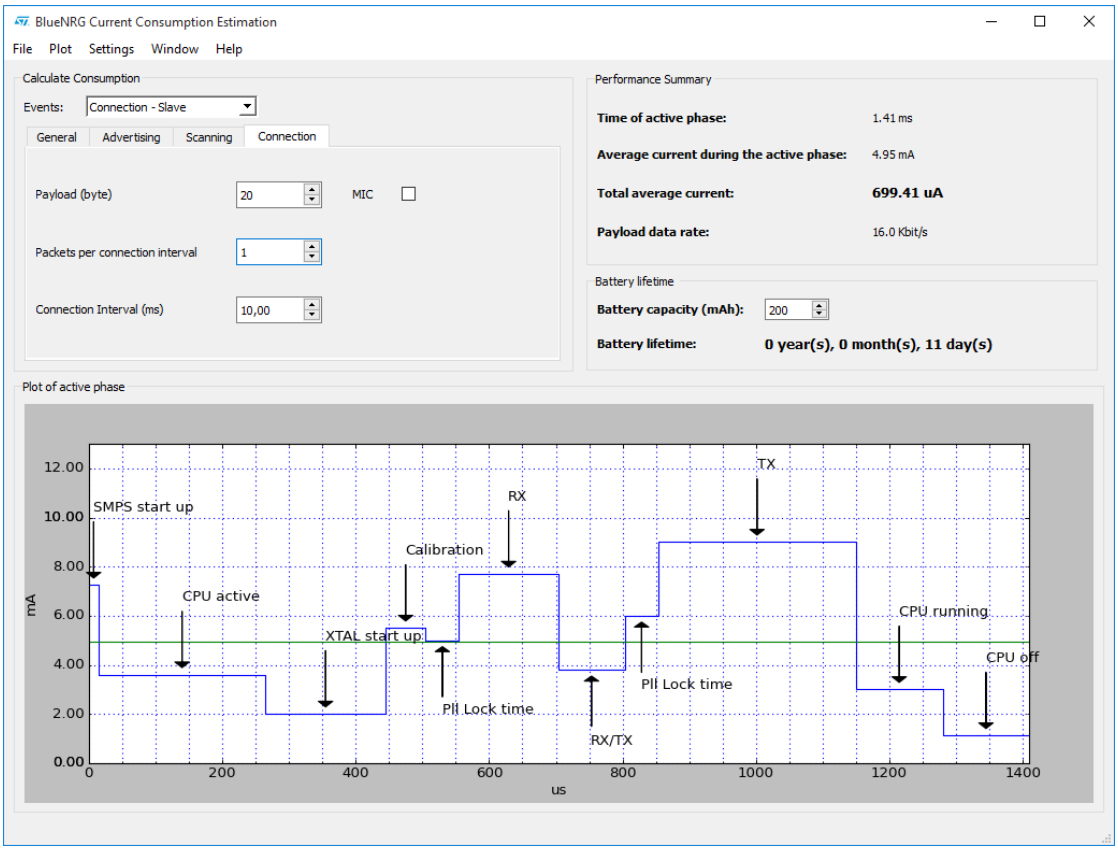

Figure 5.11.: BlueNRG Current Consumption Estimation for 20 byte payload, 1 packet per interval and 10 ms of connection interval

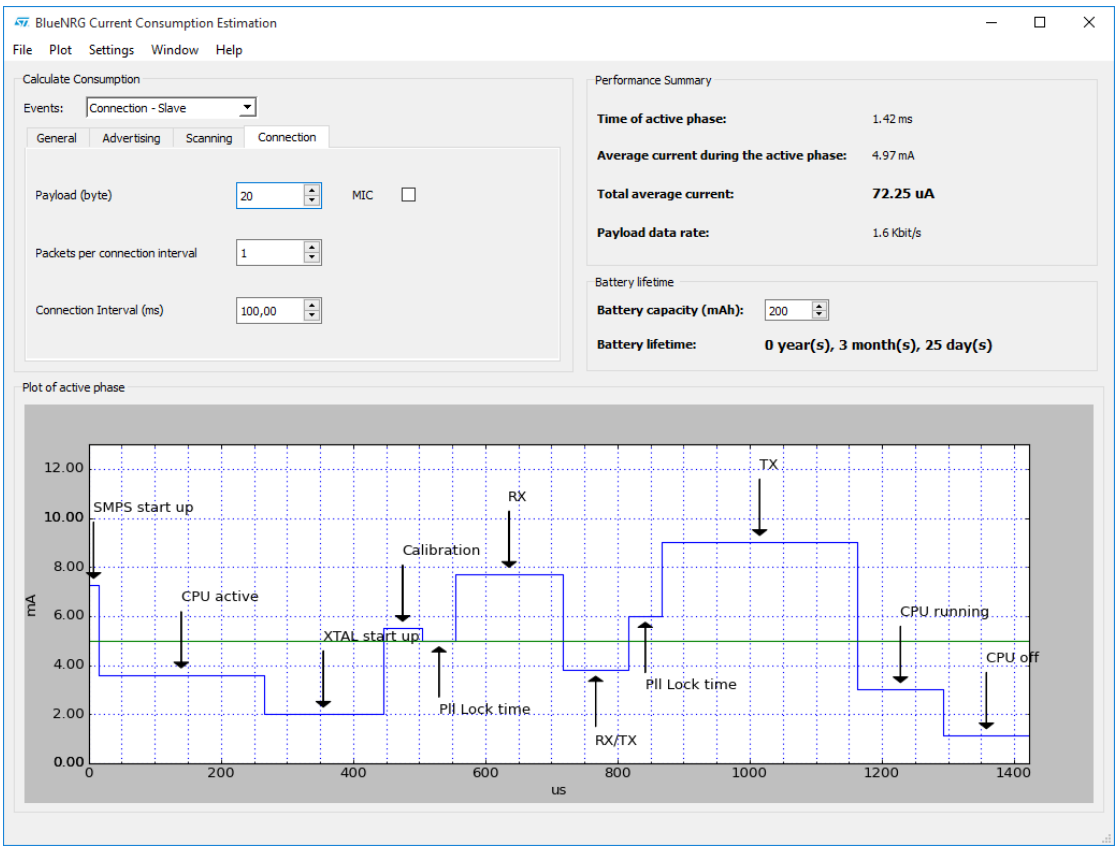

Figure 5.12.: BlueNRG Current Consumption Estimation for 20 byte payload, 1 packet per interval and 100 ms of connection interval

First we need to know if the time and current needed for one packet is dependent of the connection interval. In a connection interval there are two states, an active state and a stand-by state. In active state is when the data packets are exchanged, and in stand-by it turns off the controller of the BlueNRG and consumes a residual current of 1 mA. The Figure 5.11 shows the current and time needed in the active phase sending 1 packet of 20 bytes. We can compare it with Figure 5.12, where we can see that, the time and average current for the active phase is the same in both cases.

But one important thing must be noted, although it seems that with longer connection interval we have a lower total average current and a longer battery lifetime, it

does not actually save energy. As the important thing is data transfer, we must study the energy needed to send the actual data and not how long is the battery lifetime. If we take $ms \cdot uA$ as a energy unit, then we are using 6994 unit to transfer a packet of 20 bytes with the shorter interval and 7225 unit with the long interval. This difference is because in the longer interval we have a big gap of stand-by phase where the module is consuming a residual current.

A for splitting the packets into smaller intervals or merge them into a larger interval, first we should see if it takes more or less energy to send more packets in the same interval. The Figure 5.13 shows the relation between packets per interval and the time of the active phase. And the Figure 5.14 shows the relation between the number of packets and the energy consumed in the active phase. It can be observed a linear behaviour in both cases.

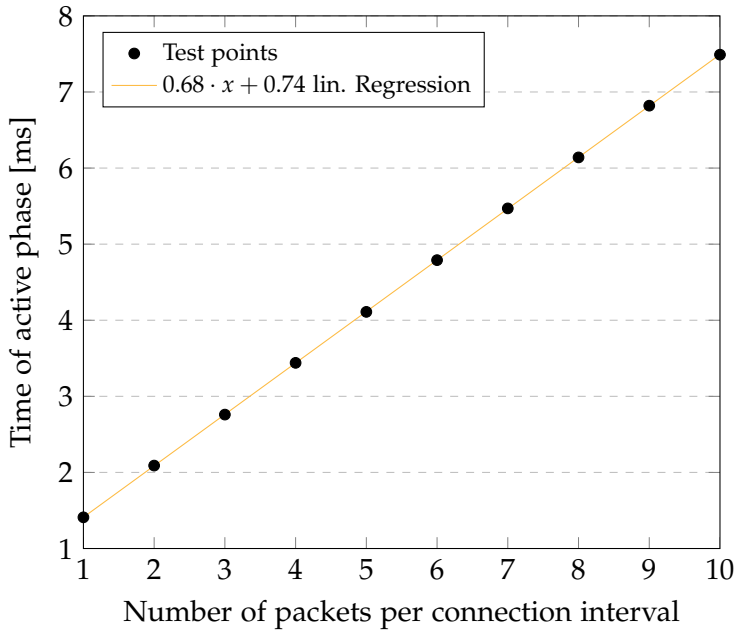

Figure 5.13.: Time active per number of packets in connection interval

This means that there is no difficulty curve as for sending packets in the same interval. Which leads to the only benefit from merging data packets into the same connection interval will be the reduction of time and energy in the initial start up and

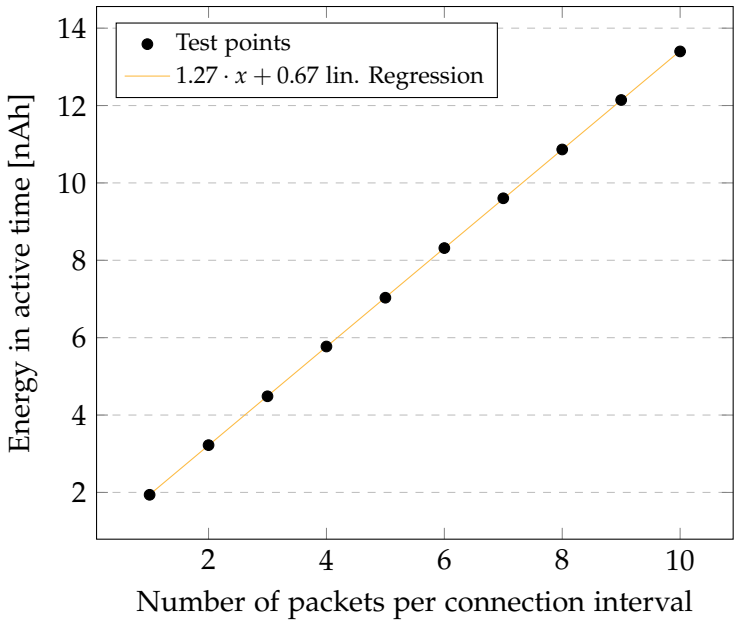

Figure 5.14.: Energy in active time per number of packets in connection interval

calibration. This can be proved in Figure 5.15 and Figure 5.16, where sending two times 10 packets in 10 ms interval is almost equal as sending once 20 packets in 20 ms. So there is no need as energy wise to merge lots of packets in a long connection interval.

In summary, the correct configuration to be energy efficient is to minimize the empty gaps of the stand-by state. That means to fill the maximum possible the connection intervals with packets, no matter if the connection interval is long or short. This at the same time will increase the bit rate.

One possible reason to set the connection interval and the number of packets is the total average current. If for some implementation reason there is a upper limit to the average current, then should set a stand-by phase long enough to lower the average current to below the limit. Another possible reason is the case of sensors and peripherals, where there is no need to transmit huge amount of data in short time, but to transmit small data in a fixed frequency.

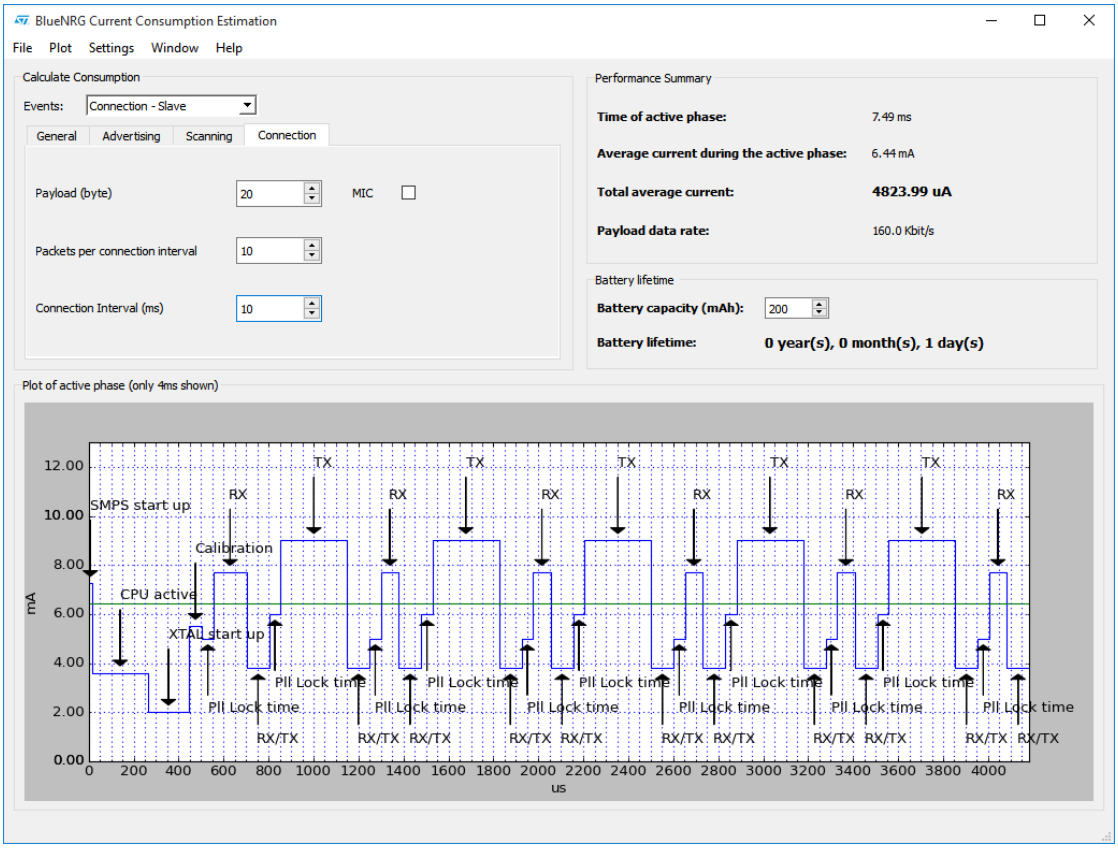

Figure 5.15.: BlueNRG Current Consumption Estimation for 20 byte payload, 10 packets per interval and 10 ms of connection interval

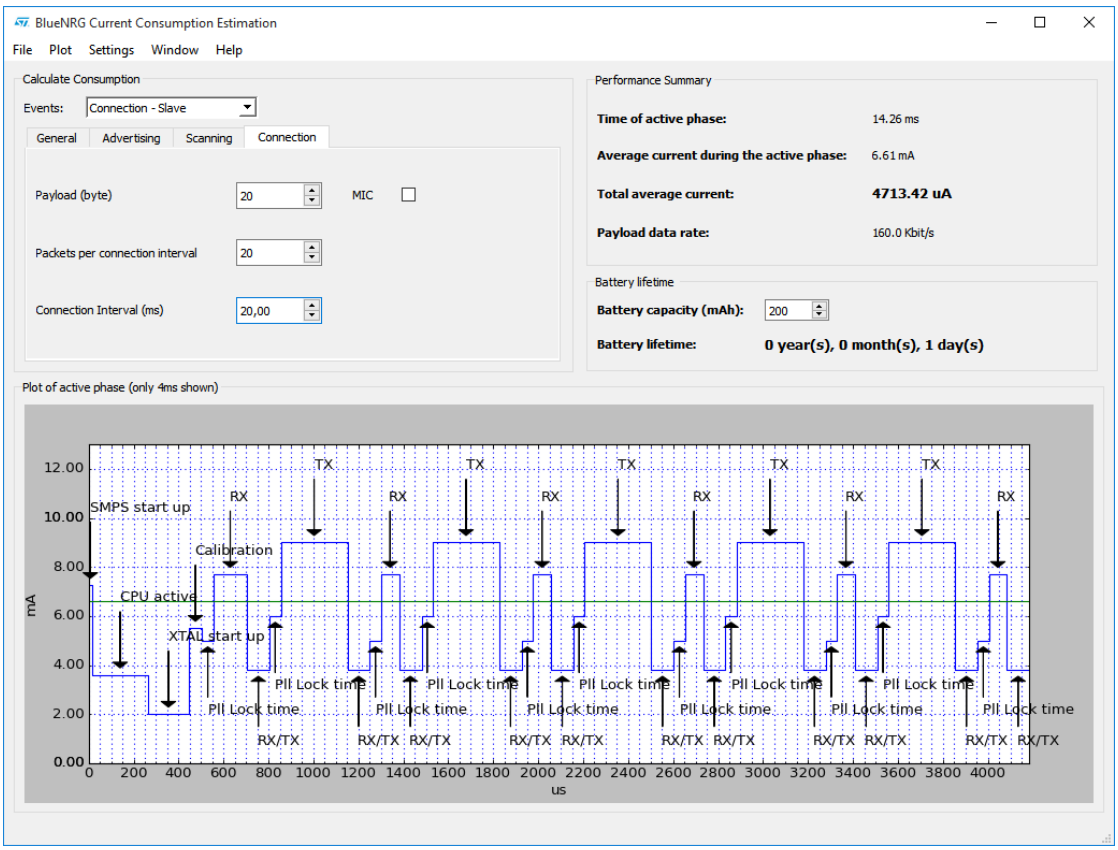

Figure 5.16.: BlueNRG Current Consumption Estimation for 20 byte payload, 20 packets per interval and 20 ms of connection interval

5.3. Final global analysis

First of all, from the data collected from the tests we can see that Bluetooth both classic and low energy have a good response from any external interferences, no matter if it is a WLAN, a RFID signal or an electrical motor. And this is because the frequency hopping and small bandwidth combined with small packets, as described in [13]. The transmitter power from both devices fall under the category of Power Class 2, with electrical and signal power near 2.5 mW (4 dBm). This power class in general implementation have a range of approximately 10 m. This can be seen in the results of

the tests, where from 10 m the quality of the communication starts to fail.

We see a acceptable performance in the closed environment with electrical equipment and metallic structures. Although there are some packets lost that need a longer time to send, overall it does not affect much on the communication quality. Those cases tend to be less than 1% with STM BlueNRG working with 8 dBm and less than 2% with Microchip RN4677 working with 2 dBm. With STM BlueNRG in -2 dBm the rate of longer transmits is slightly higher, up to 7% in the worst case scenario.

The influences from the motors can be appreciated in bigger distances as the signal start to lose power. The effects normally start to manifest from 8 m in STM when the transmit power is -2 dBm, and from 10 m with STM working with 8 dBm and Microchip. The number of packets that need longer time tend to be less than 2% with STM BlueNRG at 8 dBm, less than 4% with Microchip RN4677, and with STM working at -2 dBm can rise from 5% at 8 m up to 20% at 15 m.

About the use of different microprocessor with the STM module and the Microchip module, we can see that the STM32F103 microprocessor can achieve similar bit rate in classic mode as the STM32L151 in low energy mode. That means that the bottleneck of the Microchip in low energy mode is not from the microprocessor, but from the module itself.

We saw that for a better energy efficiency we should send more packets in a connection interval. Concerning the advertisement, we should set the advertisement and scan intervals depending on the energy specification of each device. For example, if one tool is connected to the power grid, then we can set it with lower interval, so the other device can be energy efficient.

6. Conclusion and future works

As for which Bluetooth module is best it depends more on the kind of Bluetooth application the developer want to implement. It he only need a fast and easy solution, and does not concern about performance or security, the Microchip module or other similar modules is his choice. But when he want a Bluetooth efficient, fast connections, customizable application and strong privacy and security level, then the best module is BlueNRG or similar low level Bluetooth modules.

We saw that the with Classic technology we can achieve better performances. But if we want a future application with smartphone we must adopt the Low Energy technology, as the Classic technology will consume much more energy.

In this Thesis we tested that for an optimal application using Low Energy technology, the distance between the devices must be less than 10 m. With this distance restriction in mind, maybe we are not able to connect all the machines and tools in a big production plant. But at least, for middle range communications like upgrading the Firmware, quality tests or downloading log files, the Bluetooth Low Energy solution can be a good alternative to other physical communication interfaces.

And as a alternative to other wireless communications, this must be compared individually with each possibility. As for the RFID device we used in the tests, we obtained a similar bit rate of near 10 kbits. The bit rate is similar to our actual Bluetooth bit rate but unlike Bluetooth, there are not many smartphones with full compatibility with RFID technology. So if we want to communicate our embedded systems to a smartphone, the best solution is using Bluetooth technology.

From the *BlueNRG Current Consumption Estimation Tool* we saw that the maximal bit rate for BlueNRG is approximately 230 kbits/s, and this can be achieved when

we send the maximum possible number of packets in one connection, although this was not the case in our tests. Comparing the results of the tests with the estimation tool we conclude that the STM module was transmitting between 1 and 2 packets per connection interval. The problem is that in the PCI and HCI libraries from STM there is no command used to define how many packets to send in a connection interval. And the developers of the module in STM stated that this choice of the number of packets is done internally and automatically.

So for a future research it would be interesting to see if the BlueNRG module is more optimized to arrive at this bit rate or if there is an another Bluetooth module in the market that can transmit with this speed.

Another interesting point for a future work is the possibility of use a Bluetooth packet sniffer for a better understanding of the functionality of Bluetooth communications in order to seek an optimization of the communication using Bluetooth modules.

Because of time restrictions, we could not test the performances with smartphones. It would be a interesting study to compare the performance of a smartphone with the results of this work.

A. Appendix A

A.1. Source code

```
uint64 GetTimeMs64()
{
  /* Windows */
  FILETIME ft;
  LARGE_INTEGER li;

  /* Get the amount of 100 nano seconds intervals
  elapsed since January 1, 1601 (UTC) and copy it
  * to a LARGE_INTEGER structure. */
  GetSystemTimeAsFileTime(&ft);
  li.LowPart = ft.dwLowDateTime;
  li.HighPart = ft.dwHighDateTime;

  uint64 ret = li.QuadPart;
  /* Convert from file time to UNIX epoch time. */
  ret -= 116444736000000000LL;
  /* From 100 nano seconds to 1 millisecond intervals */
  ret /= 10000;

  return ret;
}
```

Figure A.1.: Code for time stamp in milliseconds

```
void processInputData(uint8_t* RX_buffer, uint16_t RX_bytes)
{
  struct timer t;
  Timer_Set(&t, CLOCK_SECOND*10);
  BSP_LED_Toggle(LED2); // used for debugging
  while
  (
    // write the received value to TX characteristic
    aci_gatt_update_char_value
    (
      chatServHandle,
      TXCharHandle,0,
      RX_bytes,RX_buffer
    )==BLE_STATUS_INSUFFICIENT_RESOURCES
  )
  {
    // Radio is busy (buffer full).
    if(Timer_Expired(&t))
    break;
  }
}
```

Figure A.2.: Code for mirror function implemented in Nucleo Board

```
if (USART_GetITStatus(pUART->pUSART, USART_IT_RXNE) != RESET)
{
  // store the received byte to the RX buffer
  (pUART->pRX_Buffer)[pUART->uwCount] = pUSART->DR;
  // count the number of bytes received
  pUART->uwCount++;

  USART_ClearITPendingBit(pUSART, USART_IT_RXNE);

  // when we received all the bytes
  if
  (
    pUART->uwCount == pUART->Packet_Length
  )
  {
    // switch to transmit mode
    USART_ITConfig(pUART->pUSART, USART_IT_RXNE, DISABLE);
    USART_ITConfig(pUART->pUSART, USART_IT_TXE, ENABLE);
    pUART->uwCount = 0;
  }
}
```

Figure A.3.: Code of mirror function for STM32F103 for Microchip module (RX mode)

```
if (USART_GetITStatus(pUART->pUSART, USART_IT_TXE) != RESET)
{
  if (pUART->uwCount < pUART->Packet_Length)
  {
    pUSART->DR = (pUART->pubRXBuffer)[pUART->uwCount];
    pUART->uwCount++;
  }
  else
  {
    // switch back to receive mode
    USART_ITConfig(pUART->pUSART, USART_IT_TXE, DISABLE);
    USART_ITConfig(pUART->pUSART, USART_IT_RXNE, ENABLE);
    pUART->uwCount = 0;
  }
  USART_ClearITPendingBit(pUART->pUSART, USART_IT_TXE);
}
```

Figure A.4.: Code of mirror function for STM32F103 for Microchip module (TX mode)

A.2. Test graphics

Figure A.5.: Classic Mode in closed environment with 2 dB

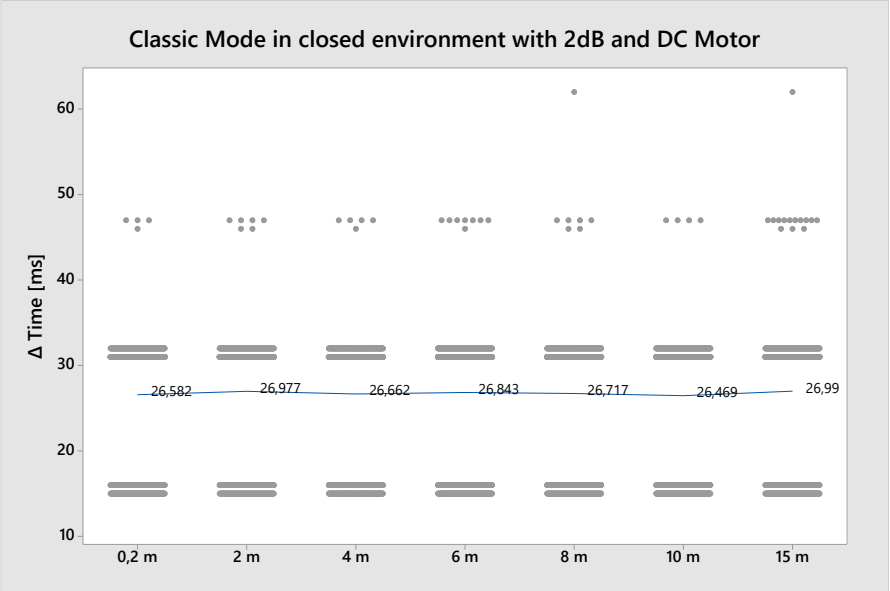

Figure A.6.: Classic Mode in closed environment with 2 dB and DC motor

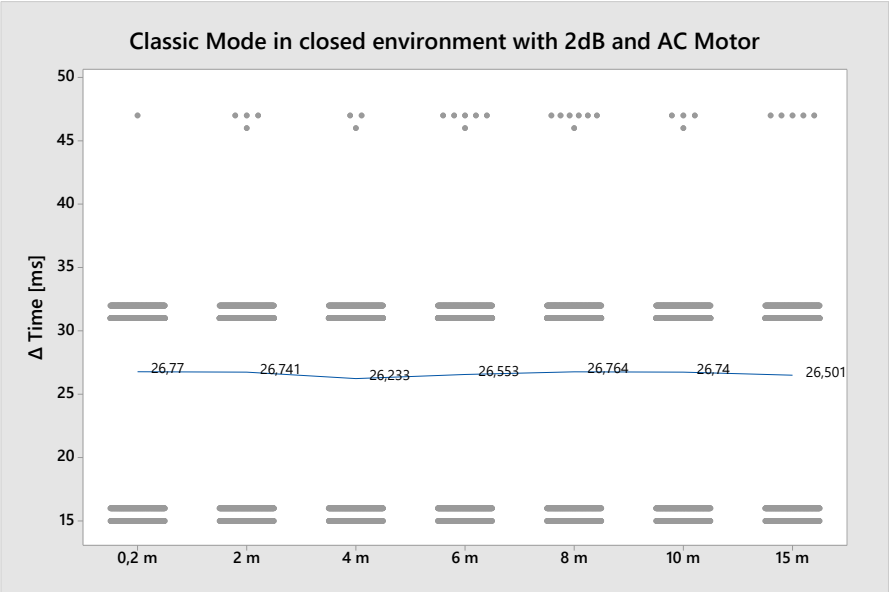

Figure A.7.: Classic Mode in closed environment with 2 dB and AC motor

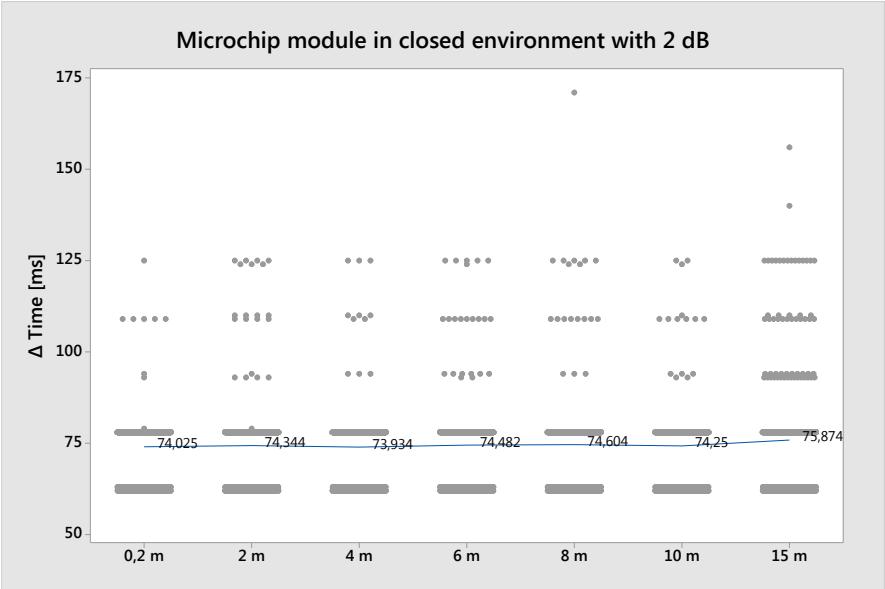

Figure A.8.: Microchip module in closed environment with 2 dB

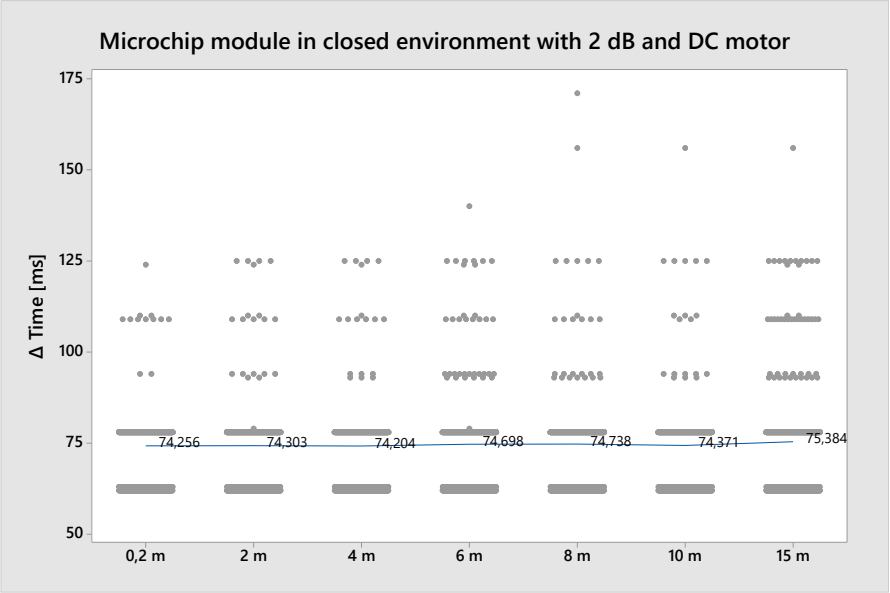

Figure A.9.: Microchip module in closed environment with 2 dB and DC motor

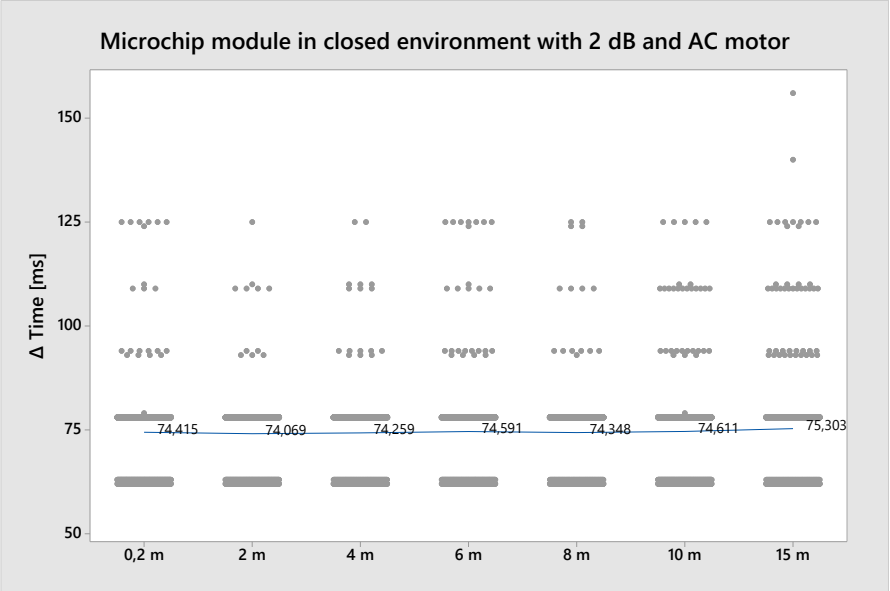

Figure A.10.: Microchip module in closed environment with 2 dB and AC motor

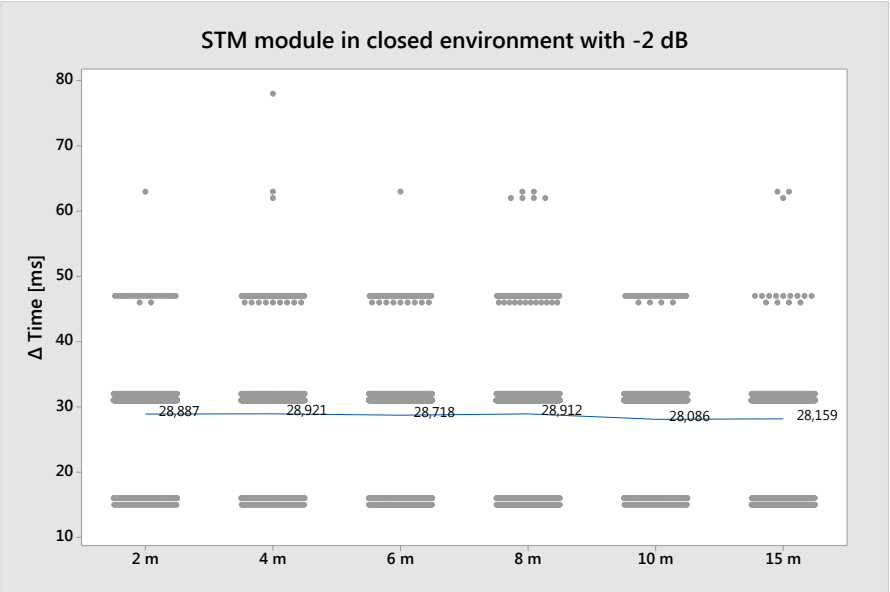

Figure A.11.: STM module in closed environment with -2 dB

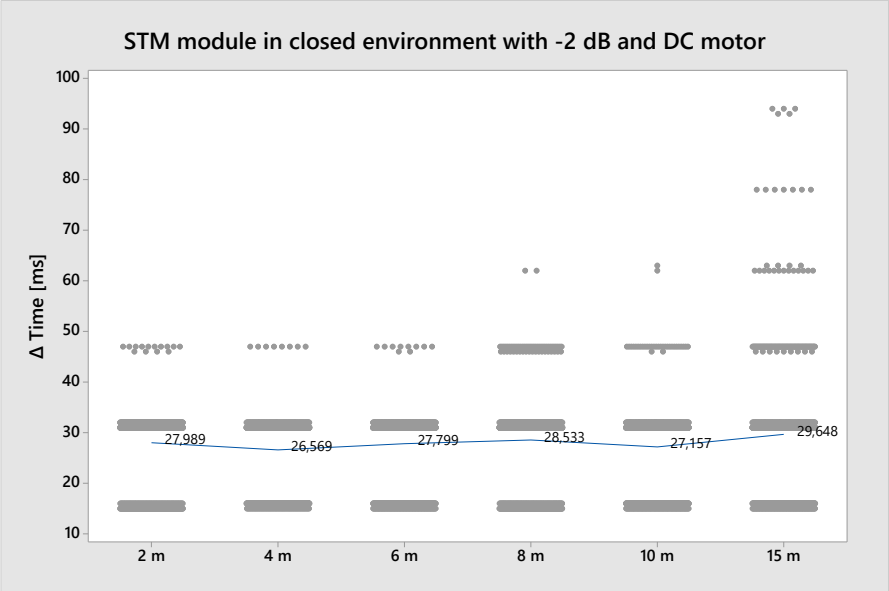

Figure A.12.: STM module in closed environment with -2 dB and DC motor

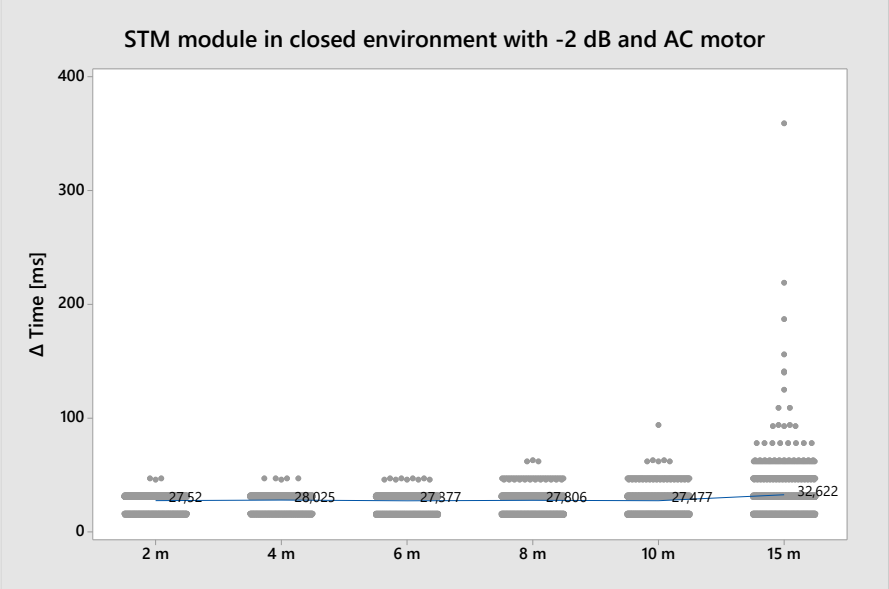

Figure A.13.: STM module in closed environment with -2 dB and AC motor

List of Figures

List of Tables

Bibliography

[1] *Our History | Bluetooth Technology Website*. URL: http://www.bluetooth.com/Pages/History-of-Bluetooth.aspx.

[2] Bluetooth SIG. *Bluetooth Core Specification v4.0*. 2010.

[3] *Service Discovery | Bluetooth Technology Special Interest Group*. URL: https://www.bluetooth.org/en-us/specification/assigned-numbers/service-discovery.

[4] *Bluetooth Development Portal*. URL: https://developer.bluetooth.org/.

[5] *16-bit UUID for Members | Bluetooth Technology Special Interest Group (need login)*. URL: https://www.bluetooth.org/en-us/specification/assigned-numbers/16-bit-uuids-members.

[6] *Bluetooth Low Energy | Bluetooth Technology Website*. URL: http://www.bluetooth.com/Pages/low-energy-tech-info.aspx.

[7] *Smartphones | Bluetooth Technology Website*. URL: http://www.bluetooth.com/Pages/Mobile-Telephony-Market.aspx.

[8] J. Banos, ed. *Testing of Bluetooth products in the industrial environment*. Vol. 4. 2002. DOI: 10.1109/IECON.2002.1182948.

[9] ETSI. *European Standard: EN 301 390 - V1.3.1*.

[10] S. Hawayek, C. Hargrove, and N. A. Bousaba, eds. *Real-time bluetooth communication between an FPGA based embedded system and an Android phone*. 2013. ISBN: 1091-0050. DOI: 10.1109/SECON.2013.6567418.

[11] D. F. Bell, J.-C. Cano, and P. Manzoni, eds. *Evaluating Bluetooth performance as the support for context-aware applications*. 2003. ISBN: 1095-2055. DOI: 10.1109/ICCCN.2003.1284192.

[12] S. Silva, S. Soares, T. Fernandes, A. Valente, and A. Moreira, eds. *Coexistence and interference tests on a Bluetooth Low Energy front-end*. 2014. DOI: 10.1109/SAI.2014.6918312.

[13] M. Fainberg and D. Goodman, eds. *Analysis of the interference between IEEE 802.11b and Bluetooth systems*. Vol. 2. 2001. ISBN: 1090-3038. DOI: 10.1109/VTC.2001.956918.

[14] Ting-Yu Lin, Yen-Ku Liu, and Yu-Chee Tseng. "An improved packet collision analysis for multi-Bluetooth piconets considering frequency-hopping guard time effect". In: *Selected Areas in Communications, IEEE Journal on* 22.10 (2004), pp. 2087–2094. ISSN: 0733-8716. DOI: 10.1109/JSAC.2004.836018.

[15] K. Morsi, Gao Qiang, and Xiong Huagang, eds. *Interference impact on throughput performance of Bluetooth scatternets under different traffic loads*. 2010.

[16] A. Rahim and A. Finger, eds. *Estimation of the impact of electromagnetic and co-channel interference on the Bluetooth receiver*. 2005. DOI: 10.1109/CANET.2005.1598185.

[17] Z. Q. Zhu and D. Howe, eds. *Electromagnetic noise radiated by brushless permanent magnet DC drives: Electrical Machines and Drives, 1993. Sixth International Conference on (Conf. Publ. No. 376)*. 1993.

[18] Hao Chen, Lingguo Cheng, Xiaohui Qiu, and Yang Zhao, eds. *Conductive EMI Noise Measurement for Switched Reluctance Drive: Electromagnetic Compatibility, 2009 20th International Zurich Symposium on*. 2009. DOI: 10.1109/EMCZUR.2009.4783435.

[19] Qiu Xiaohui, Lu Xiaoquan, Zhao Yang, Dong Yinghua, Jiang Ningqiu, Rong Rong, and Yang Qiutong, eds. *Conductive EMI noise analysis for large power switched reluctance machine: Electric Power and Energy Conference (EPEC), 2010 IEEE*. 2010. DOI: 10.1109/EPEC.2010.5697190.

[20] M. Tan and K. A. Masagca, eds. *An Investigation of Bluetooth Security Threats*. 2011. DOI: 10.1109/ICISA.2011.5772388.

[21] JunFeng Xu, Tao Zhang, Dong Lin, Ye Mao, Xiaonan Liu, Shiwu Chen, Shuai Shao, Bin Tian, and Shengwei Yi, eds. *Pairing and Authentication Security Technologies in Low-Power Bluetooth*. 2013. DOI: 10.1109/GreenCom-iThings-CPSCom.2013.185.

[22] S. Sandhya and K. A. S. Devi, eds. *Analysis of Bluetooth threats and v4.0 security features*. 2012. DOI: 10.1109/ICCCA.2012.6179149.

[23] A. S. Diallo, W. F. M. Al-Khateeb, R. F. Olanrewaju, and F. Sado, eds. *A Secure Authentication Scheme for Bluetooth Connection*. 2014. DOI: 10.1109/ICCCE.2014.29.

[24] C. Gomez, J. Oller, and J. Paradells. "Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology". In: *Sensors* 12.12 (2012), pp. 11734–11753. ISSN: 1424-8220. DOI: 10.3390/s120911734.

[25] Jia Liu, Canfeng Chen, Yan Ma, and Ying Xu, eds. *Energy Analysis of Device Discovery for Bluetooth Low Energy*. 2013. ISBN: 1090-3038. DOI: 10.1109/VTCFall.2013.6692181.

[26] Microchip Technology Inc. *RN4677 Bluetooth 4.0 Dual Mode Module Datasheet*. 2015.

[27] STMicroelectronic. *BlueNRG Bluetooth LE wireless network processor Datasheet*. 2014.

[28] STMicroelectronic. *X-NUCLEO-IDB04A1: Bluetooth low energy expansion board based on BlueNRG for STM32 Nucleo*. 2014.

[29] STMicroelectronic. *NUCLEO-XXXXRX: STM32 Nucleo-64 boards*. 2015.

[30] STMicroelectronic. *AN4559 Application note: Developer's guide to creating Bluetooth low energy applications using STM32 Nucleo and BlueNRG*. 2014.

[31] STMicroelectronic. *STEVAL-IDB002V1: Bluetooth SMART board based on the BlueNRG low energy network processor*. 2013.

[32] STMicroelectronic. *STEVAL-IDB003V1: A BlueNRG IC based Bluetooth Smart USB dongle*. 2014.

[33] STMicroelectronic. *AN4642 Application note: Overview of the BLE Profiles application for X-CUBE-BLE1, expansion for STM32Cube*. 2014.

[34] STMicroelectronic. *UM1755 User manual: BlueNRG Bluetooth LE stack application command interface (ACI)*. 2015.